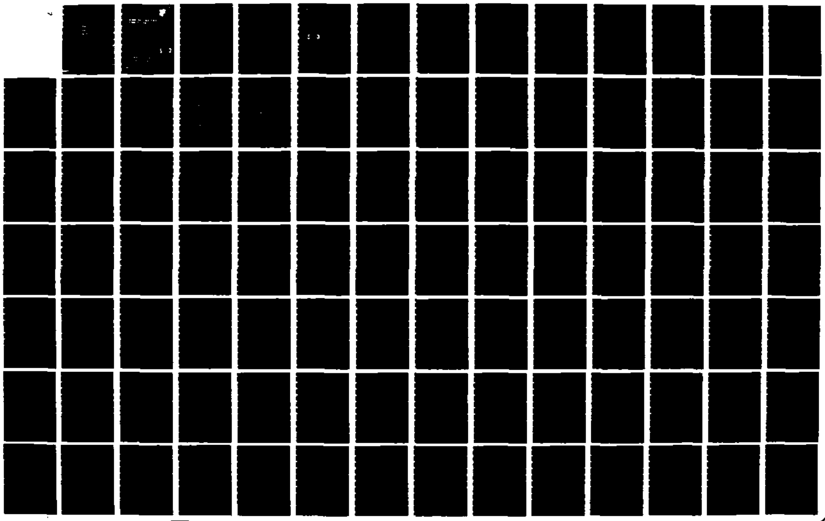
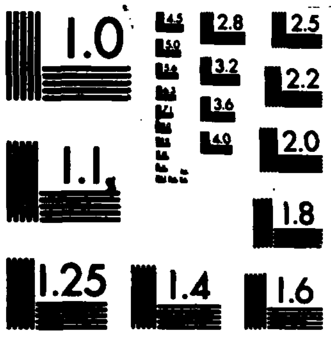
AD-A137 955 SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS 1/2
VOLUME 1(U) BOEING AEROSPACE CO SEATTLE WA
T P BOWEN ET AL. JUL 83 RADC-TR-83-175-VOL-1
UNCLASSIFIED F30602-80-C-0330 F/G 9/2 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0

12

AD A137955

RADC-TR-83-175, Vol I (of three)
Final Technical Report
July 1983

SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS

Boeing Aerospace Company

Thomas P. Bowen, Jonathan V. Post, Juitien Tsai, P. Edward Presson and Robert L. Schmidt

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441

RADC-TR-83-175, Vol I (of three) has been reviewed and is approved for publication.

APPROVED: [illegible]

JOSEPH P. CAVANO
Project Engineer

APPROVED: [illegible]

RONALD S. RAPOSO
Acting Chief, Command & Control Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-83-175, Vol I (of two)	2. GOVT ACCESSION NO. AD-A137955	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report October 80 - March 83
		6. PERFORMING ORG. REPORT NUMBER N/A
7. AUTHOR(s) Thomas P. Bowen, Jonathan V. Post, Juitien Tsai, P. Edward Presson, Robert L. Schmidt		8. CONTRACT OR GRANT NUMBER(s) F30602-80-C-0330
9. PERFORMING ORGANIZATION NAME AND ADDRESS Boeing Aerospace Company PO Box 3999 Seattle WA 98124		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 55812030
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (COEE) Griffiss AFB NY 13441		12. REPORT DATE July 1983
		13. NUMBER OF PAGES 138
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Joseph P. Cavano (COEE)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software Quality
Software Metrics
Software Measurement
Software Survivability
Software Expandability

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Software metrics (or measurements) which are used to indicate and predict levels of software quality were extended from previous research to include considerations for distributed computing systems. Aspects of the products of software life-cycle activities which could affect the quality levels of software, and metrics to measure them, were identified. Two new quality factors, survivability and expandability, were validated. A Guidebook for Software Quality Measurement was produced to aid in setting quality goals, applying metric measurements, and making quality level assessments. New metrics for interoperability and reusability were also included in the guidebook.

Accession For	
NTIS GRA&I	✓
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

PREFACE

This document is the final technical report (CDRL A003) for the Quality Metrics for Distributed Systems contract, number F30602-80-C-0330. The contract was performed for Rome Air Development Center (RADC) to provide methodology and technical guidance on software quality metrics to Air Force Software acquisitions managers.

This report consists of three volumes as follows:

Volume I	Software Quality Measurement for Distributed Systems - Final Report
Volume II	Guidebook for Software Quality Measurement
Volume III	Distributed Computing Systems: Impact on Software Quality

The objective of this contract was to conduct exploratory development of techniques to measure system quality with a perspective on both software and hardware from a life cycle viewpoint. The effort was expected to develop and validate metrics for software quality on networked computers and distributed systems; i.e., systems whose functions may be tightly distributed over microprocessors or specialized devices such as data base machines. At the same time, the effects hardware has on software was to be studied, as well as the trade-offs between hardware, firmware, and software. The results of this research are reported in ~~Volume I.~~ this volume.

Volume II describes the application of quality metrics to distributed systems and provides guidance for AF acquisition managers. The guidebook provides guidance for specifying and measuring the desired level of quality in a software product.

Volume III describes a qualitative study of distributed system characteristics, reasons for selection, design strategies, topologies, scenarios, and trade-offs. These analyses led to the changes in the Framework shown in Volume I, and to the validation of models.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

SECTION 1.0
EXECUTIVE SUMMARY

1.1 OBJECTIVES OF RESEARCH

This work was performed under a research contract (F30602-80-C-0330) for Rome Air Development Center, Griffiss AFB, NY. The object of this effort was to develop techniques which can be used to measure distributed system software quality. The study looks at both hardware and software from a life cycle viewpoint, studies the effect that hardware or operating environments, i.e., distributed systems, have on software quality, and studies tradeoffs between hardware, firmware and software. This study was conducted to develop and validate proposed metrics for software quality on netwo .d computers and distributed systems. This effort was also conducted to expand and r e the software quality measurement framework defined in prior Government (R C) contracted work; Factors in Software Quality, F30602-76-C-0417 and Software Qu .r Metrics Enhancement, F30602-78-C-0216.

1.2 BACKGROUND

Rome Air Development Center (RADC) has been pursuing a program intended to achieve better control of software quality since 1976. This program has been seeking to identify the key issues and provide a valid methodology for specifying and measuring software quality requirements for software developed for major AF weapon systems.

In 1976 RADC and the Electronic Systems Division (ESD) sponsored an effort which defined a set of eleven user-oriented characteristics or quality factors (correctness, efficiency, integrity, usability, testability, flexibility, reusability, maintainability, reliability, portability, and interoperability) which extended throughout the software life-cycle. This effort established a hierarchical software quality measurement framework as shown in Figure 1.1-1. The user-oriented factors, for use by acquisition managers to

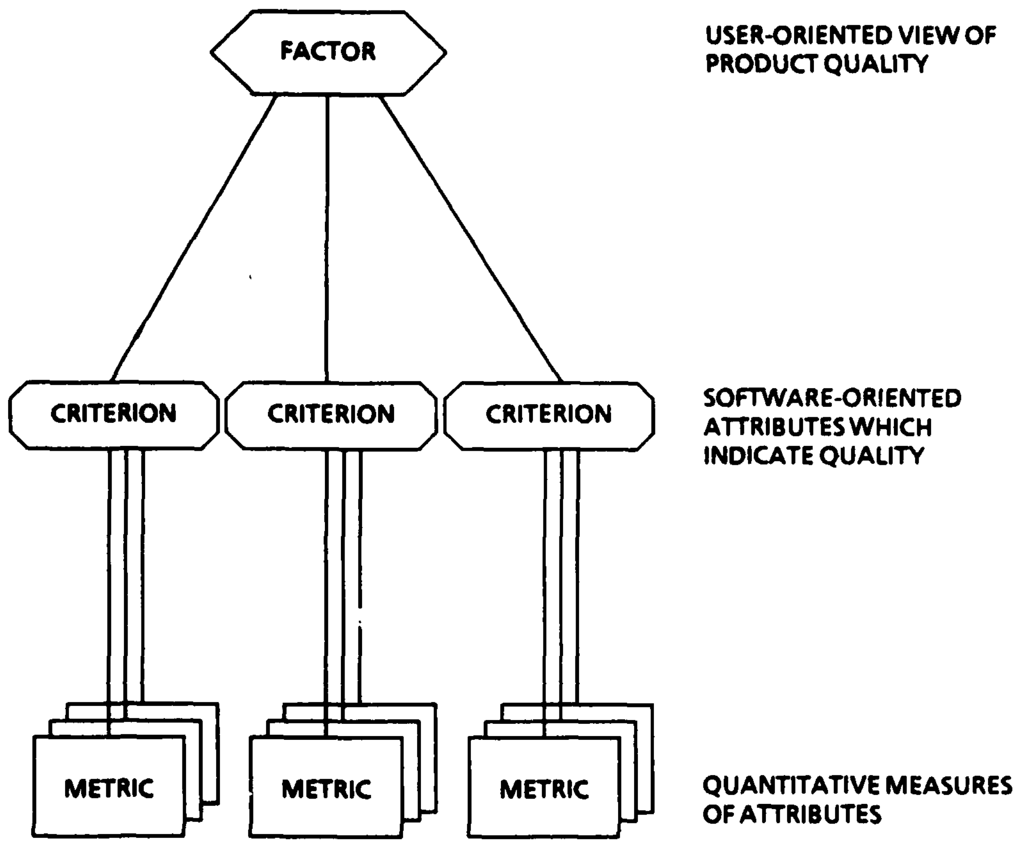

FACTOR
USER-ORIENTED VIEW OF PRODUCT QUALITY
CRITERION
CRITERION
CRITERION
SOFTWARE-ORIENTED ATTRIBUTES WHICH INDICATE QUALITY
METRIC
METRIC
METRIC
QUANTITATIVE MEASURES OF ATTRIBUTES

Figure 1.1-1 Software Quality Model/Framework

specify quality requirements, are at the top level. The software oriented criteria (attributes which indicate quality) and software metrics (quantitative measures of attributes) are at the second and third levels respectively. The metrics represent the most detailed level of the framework and completely define quality in terms of measurable elements. Taken collectively, this hierarchy formed the basis of a model for predicting and controlling software quality. This research was performed under contract F30602-76-C-0417, and was described in technical report RADC-TR-77-369, Factors in Software Quality.

In 1978, RADC and the US Army Computer Systems Command sponsored additional research to enhance this framework under contract F30602-78-C-0216, Software Quality Metrics Enhancement. The results of this effort were reported in technical report RADC-TR-80-109, Volume I, Software Quality Metrics Enhancement and Volume II, Software Quality Measurement Manual. The manual provides methodology to assist the AF acquisition manager in describing to a contractor what quality factors the manager considers the most important.

In 1979, RADC and the US Army Computer Systems Command issued contract F30602-79-C-0267 to develop an Automated Quality Measurement Tool for the H6180/GSOS Computer system. The purpose of this tool was to automate the collection of specific metric data, and to provide quality measurement assessments. This tool was delivered to the Air Force in September 1981.

In 1980, RADC sponsored further research of the software factors of interoperability and reusability (Contract F30602-80-C-0265). The objective was to enhance the Software Metric Model by incorporating new findings for these two factors which had not been extensively studied in prior research contracts. The contract resulted in adding new criteria and metrics for these factors. The results of this effort are incorporated in Volume II of this report (Software Quality Measurements Manual).

In 1980, RADC also sponsored this research contract to extend the quality measurement framework to distributed systems and to transition the information into a form useful to the AF software acquisition manager. The prior research focused primarily on the software subsystem and largely ignored the total system aspects such as the computing hardware, operating system and communications network. The increasing demand and importance of distributed systems in future defense systems created a need to extend the framework to address system quality concerns which affect the emphasis and meaning of software quality. The system quality concerns changed the emphasis of a software quality factor, criterion or metric and required modifications and additions to the software quality framework.

It will be increasingly important to understand distributed computer systems. Some of their characteristics will emerge more extensively in future configurations. One characteristic peculiar to distributed systems, and of importance in the 1980's, is geographic dispersion. The extent to which computers within a distributed system can be physically displaced from each other, range from the centimeter to the multi-thousand-kilometer. Computers will be "tightly-coupled" over intercontinental distances by fiber-optics technology currently under research. Interconnection of even a very small percentage of available computers will be able to form distributed systems of complexity beyond those of today, since by 1999 there will be on the order of one billion computers in the world.

1.3 TECHNICAL APPROACH

The approach to this problem was to use the previous work accomplished by RADC (see section 1.2) as well as previous Boeing software quality metrics research as a baseline. The technical approach was divided into a series of 9 tasks in order to accomplish the objectives of section 1.1 and the requirements of section 4 of the contract statement of work.

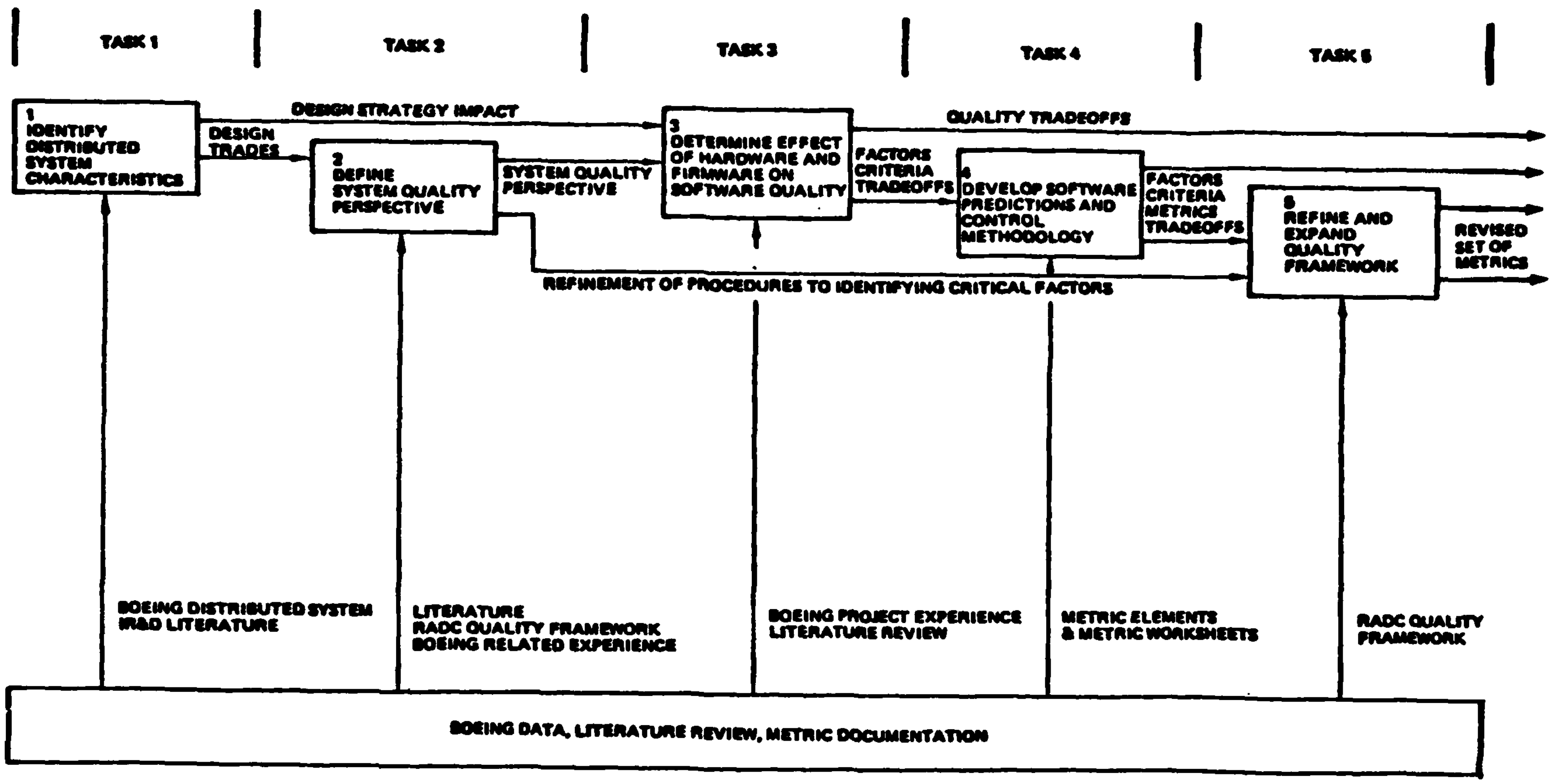
Task 1:	Identify distributed system characteristics
Task 2:	Define system quality perspective
Task 3:	Determine effect of H/W and F/W on S/W quality
Task 4:	Develop software prediction and control methodology
Task 5:	Refine and expand quality framework
Task 6:	Select quality metrics for validation
Task 7:	Develop scenarios and collect data
Task 8:	Validate metrics
Task 9:	Integrate results into a guidebook

Figure 1.3-1 summarizes the interrelationships among these tasks.

In Task 1, the characteristics that distinguish distributed systems from uniprocessors and centralized processors were identified. Distributed systems were analyzed from several standpoints, including: rationale for selecting a distributed system architecture; characterization of the distributed system design philosophy; and impact of hardware architecture on distributed system quality factors. The results served as input to Tasks 2 and 3, are reported in Volume III of this report, and are summarized in section 2.3 of this volume.

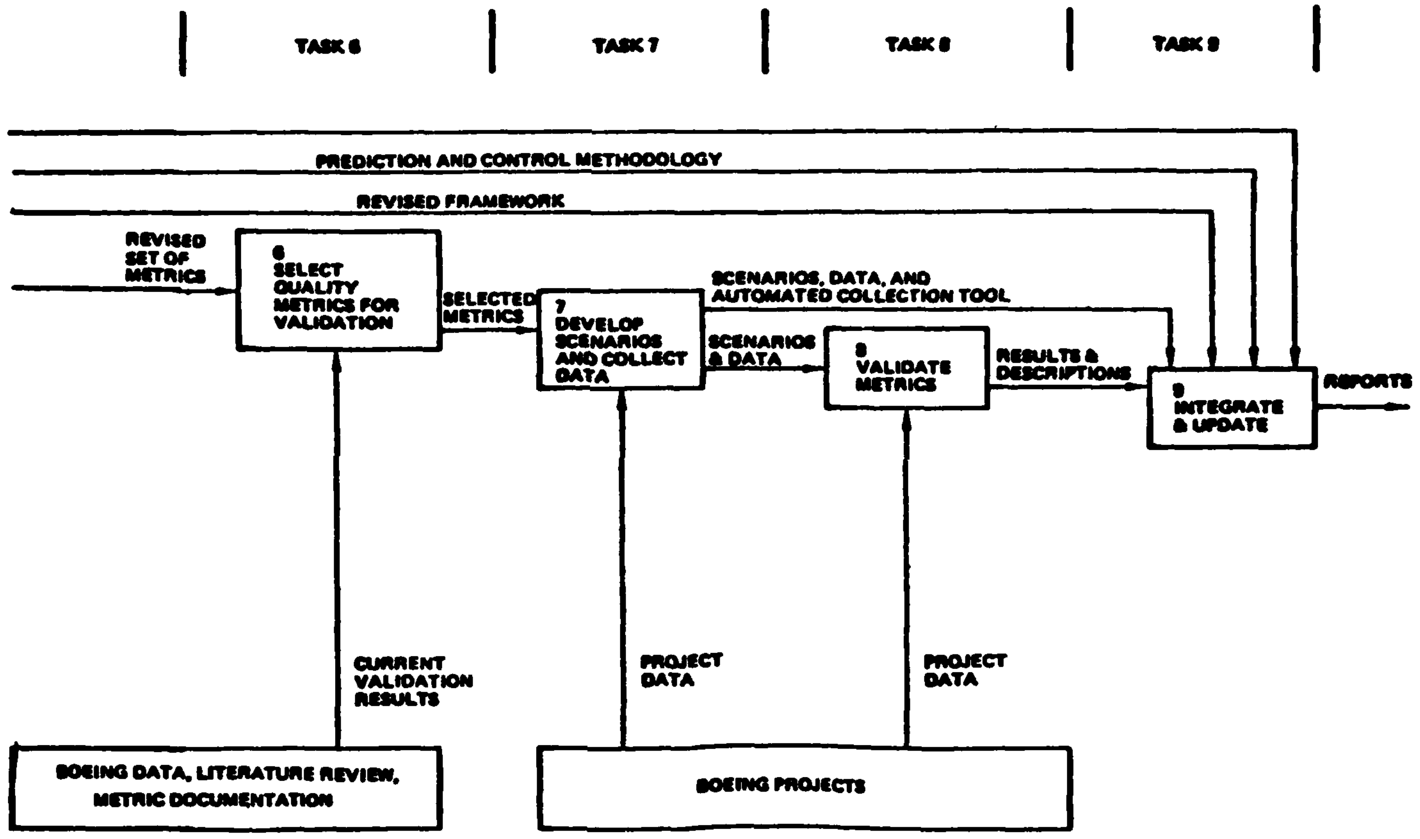
In Task 2, a framework for including software in the process of allocating system requirements and quality goals was developed. Distributed system quality factors were developed and related to software quality factors and the impact of system quality was assessed. The results provided input to Tasks 3 and 5 and are reported in section 2.2.

Figure 1.3-1: *Quality Metrics for Distributed Systems Task Flow*



TASK 1
TASK 2
TASK 3
TASK 4
TASK 5
1 IDENTIFY DISTRIBUTED SYSTEM CHARACTERISTICS
DESIGN STRATEGY IMPACT
DESIGN TRADES
2 DEFINE SYSTEM QUALITY PERSPECTIVE
SYSTEM QUALITY PERSPECTIVE
3 DETERMINE EFFECT OF HARDWARE AND FIRMWARE ON SOFTWARE QUALITY
QUALITY TRADEOFFS
FACTORS CRITERIA TRADEOFFS
4 DEVELOP SOFTWARE PREDICTIONS AND CONTROL METHODOLOGY
FACTORS CRITERIA METRICS TRADEOFFS
5 REFINE AND EXPAND QUALITY FRAMEWORK
REVISED SET OF METRICS
REFINEMENT OF PROCEDURES TO IDENTIFYING CRITICAL FACTORS
BOEING DISTRIBUTED SYSTEM IR&D LITERATURE
LITERATURE RADC QUALITY FRAMEWORK BOEING RELATED EXPERIENCE
BOEING PROJECT EXPERIENCE LITERATURE REVIEW
METRIC ELEMENTS & METRIC WORKSHEETS
RADC QUALITY FRAMEWORK
BOEING DATA, LITERATURE REVIEW, METRIC DOCUMENTATION

Figure 1.3-1: *Quality Metrics for Distributed Systems Task Flow*


TASK 6
TASK 7
TASK 8
TASK 9
PREDICTION AND CONTROL METHODOLOGY
REVISED FRAMEWORK
REVISED SET OF METRICS
6 SELECT QUALITY METRICS FOR VALIDATION
SELECTED METRICS
7 DEVELOP SCENARIOS AND COLLECT DATA
SCENARIOS, DATA, AND AUTOMATED COLLECTION TOOL
SCENARIOS & DATA
8 VALIDATE METRICS
RESULTS & DESCRIPTIONS
9 INTEGRATE & UPDATE
REPORTS
CURRENT VALIDATION RESULTS
PROJECT DATA
PROJECT DATA
BOEING DATA, LITERATURE REVIEW, METRIC DOCUMENTATION
BOEING PROJECTS

In Task 3, the distributed system design process was analyzed from several points of view, looking at issues that arise in allocating system quality requirements to hardware and software. Issues that arise between acquisition managers - system/software, software/ hardware and between acquisition managers and contractors were addressed. Distributed system trade-offs among hardware, firmware and software were identified. Task 3 results provide input to Task 4 and are reported in section 6.3 and 7.2 of Volume III.

In Task 4, prediction and control methodology was developed for analyzing a distributed system in terms of software quality. The methodology includes assessment of the influence of system elements to enable design latitudes in achieving desired quality of software. The results are input to Task 5 for updating the framework and are reported in Sections 3, 4, and 5 of Volume III.

In Task 5, the software quality measurement framework documented in prior RADC Technical Reports, "Software Quality Metric Enhancement" and "Software Quality Measurement Manual", (RADC-TR-80-109, Vol. I & II) was refined, expanded, and improved for factors relevant to distributed systems. The metric worksheets and tables were modified based on the results of prior Tasks. These results are used in Task 6 to aid in selecting metrics to be validated and are reported in section 3.0 of Volume II.

In Task 6, those metrics that will make the greatest contribution to validating the framework were selected based on the number of criteria to which the metric is applicable, the expected sensitivity of quality factors to the metric, the results of previous validation efforts, the value of the metric in establishing trade-offs between quality factors and between HW/FW/SW, and the results of pilot testing. Selected metrics are input to Task 7 and are reported in section 5.0.

In Task 7, scenarios for data collection were identified and designed, and data was collected to perform validation. Data was collected from Boeing projects with distributed systems. The potential for automated data collection was assessed. These scenarios were used to collect data in Task 8 and the results of this Task are reported in section 5.2.

In Task 8, the validation techniques used in RADC-TR-77-369 were reviewed and compared against other methods and refinements were made. Methods such as regression analysis without forcing the line through the origin and use of stepwise multiple regression were investigated. The metric data was collected from selected Boeing projects and the results evaluated to update the metric worksheets and validate the methodology. The results of this validation effort are reported in section 5.3 and 5.4.

In Task 9, all of the results of the other tasks were integrated incrementally to prepare contract reports and the guidebook. Volume I, Software Quality Measurement for Distributed Systems, includes an Executive Summary, the Factor/Criterion/Metric Framework, and the Statistical Validation. The guidebook describing how to use quality metrics for distributed systems is contained in Volume II, Guidebook for Software Quality Measurement. Volume III, Distributed Computing Systems: Impact on Software Quality, identifies the characteristics of distributed systems, provides an historical overview, analyzes the reasons for selection of distributed systems, classifies design strategies, provides a taxonomy of architecture topologies, presents scenarios, and analyzes hardware/software/firmware tradeoffs.

1.4 SUMMARY OF ACCOMPLISHMENTS

The objective of this contract was to develop techniques to measure distributed system quality with a perspective on both hardware and software from a life-cycle viewpoint. The approach used in accomplishing this objective was to:

1) extend the factor/criteria/metric framework of McCall et.al. to distributed computing systems;
2) define and validate new factors, criteria, and metrics to meet the unique characteristics of such systems;
3) develop design guidelines and tradeoffs for choosing between quality factors;
4) incorporate techniques into a guidebook for implementation.

The original factor/criteria/metric framework of McCall et.al. is described in section 2.1. The attributes of distributed systems, and their impact on the factor/criteria/metric framework, are explored at length in Volume III. The transition from the original (McCall) framework to the new framework is outlined in section 2.3, and the new factors, criteria, and metrics are listed in section 2.2. Design guidelines and tradeoffs for choosing between quality factors are given in section 3.1 of Volume II. Volume II is the guidebook for implementation.

1.4.1 Framework Changes

The following paragraphs summarize changes in the elements of the metric framework.

New and Modified Framework Elements

Two new quality factors were defined, survivability and expandability, and the quality factor testability was changed to verifiability, a more general term. Five new criteria were added to the framework, and eleven former criteria were modified and condensed into seven criteria. Eleven new metrics and sixty-seven new metric elements were defined.

The two new quality factors are defined as follows:

Survivability - probability that the software will continue to perform or support critical functions when a portion of the system is inoperable; and

Expandability - effort to increase software capability or performance by enhancing current functions or adding new functions/data.

Survivability is of considerable importance to distributed systems, but was legitimately overlooked by McCall et.al. in their framework for uniprocessors. From a user standpoint, a uniprocessor is either "up" (operational) or "down" (non-operational), whereas a distributed system must be designed to cope with situations where a portion of the system is inoperable. The criteria for survivability are autonomy, distributedness, anomaly management, modularity, and reconfigurability.

Expandability, as a factor, recognizes the way in which computer systems grow during their life-cycle. Also applicable to uniprocessors, Expandability measures the relative ease with which this growth is possible. The criteria for expandability are virtuality, generality, modularity, augmentability, specificity, and simplicity.

The details of the framework changes are presented in Section 2.2, Framework Enhancements. The details of the new framework are presented in Section 3.0, Distributed System Quality Metrics Framework.

Framework Elements Proposed and Dropped

One quality factor, evolvability, was dropped from the new framework because of its similarity to expandability. Six criteria were dropped from the new framework because of their similarity to other criteria or because of lack of metrics to support them. No metrics were dropped, but twenty-six metric elements were proposed and dropped either because of their overlap with other metric elements or because of their vagueness.

The details of the framework elements which were proposed and dropped are presented in Section 4.0, Transitional Study.

1.4.2 Design Guidelines and Tradeoff Analyses

New design guidelines were added to support tradeoff analyses for the situations where requirements for high quality levels for two quality factors would conflict; i.e., where specifying a high quality level for one factor would naturally result in a low quality level for another factor. New guidelines were added for tradeoffs between survivability and five other factors - efficiency, integrity, flexibility, portability, and reusability. New guidelines were added for tradeoffs between expandability and three other factors - reliability, efficiency, and integrity. New guidelines were also added for tradeoffs between reliability and two other factors - efficiency and flexibility.

The complete set of design guidelines for tradeoff analyses is presented in Section 3.1.1, Software Quality Factors.

1.4.3 Quality Metrics Guidebook

The guidebook for quality metrics is Volume II of this report. It represents a substantial update to the former handbook - Software Quality Measurement Manual, RADC-TR-80-109. The guidebook includes updates for distributed systems from this contract and incorporates results from contract F30602-80-C-0265, Software Interoperability and Reusability. The metric worksheets and metric tables are now appendices to Volume II and are considerably revised, extended and reorganized for greater convenience of use and updating. The worksheets are now more parallel in structure to ease the data gathering task. The metric tables are now organized alphabetically by criteria name.

1.4.4 Data Collection, Analyses, and Validation

Data was collected and analyzed for the two new quality factors - survivability and expandability. Metric data and quality level rating estimates were gathered from four projects with distributed computing systems: UDACS (Universal Display and Control System), E-3A (Airborne Warning and Control System), B-1 Avionics (for the B-1 bomber), and MPRT (Morgantown Personal Rapid Transit System).

Correlations between quality ratings and metric values show a positive relationship and therefore support the addition of the expandability and survivability quality factors. This result, and the fact the data collection and analysis methodology was used successfully, also supports the applicability of the software metrics technology to distributed computing systems. The details of this effort are presented in Section 5.0, Validation.

1.5 CONCLUSIONS

The work performed under this contract, the work performed under the Interoperability and Reusability contract, and the work performed by other contractors are part of a continuing technological trend. The need for more thorough measurement of software and system quality is being met by a more detailed and complete methodology. Distributed embedded computer systems are more complex than uniprocessor systems, and the applicability of the factor/criterion/metric methodology to these systems is an accomplishment which supports the continued use of that methodology. Among other benefits, this enables the tradeoffs between hardware, software, and firmware to be made more objectively than in the past; this may improve cost, schedule, and optimization. The basis of the methodology is to measure as many attributes of the system as are relevant, and to appropriately weight these measurements. This is a complex task, but is more realistic and effective than to search for a small number of metrics which somehow describe the entire system.

SECTION 2.0
SOFTWARE QUALITY FRAMEWORK ENHANCEMENTS

This section identifies enhancements made to the original software quality framework. The original framework was developed under previous RADC contracts (F30602-76-C-0417 and F30602-78-C-0216) and was used as a baseline for expanding the framework to include distributed systems. The following paragraphs highlight the initial framework, describe framework enhancements for distributed systems, and reference the research that led t changes and additions to the framework. Section 3.0 describes the complete quality metrics framework which was developed for distributed systems. A complete description of the original quality framework can be found in RADC-TR-80-109.

2.1 INITIAL FRAMEWORK

This section identifies the framework which was developed under previous contracts and used as a baseline for enhancements. The quality model and separate framework elements are discussed.

2.1.1 Quality Model

A simple model was developed for viewing software quality which is highly flexible. This model was shown in Figure 1.1-1. A hierarchical relationship is shown between a quality factor, quality criteria, and quality metrics. Quality factors are top-level views of quality and represent concerns of the acquisition manager or product user. Criteria are software-oriented attributes. Criteria are a more detailed representation of what is meant by quality and are technically oriented. A metric is a specific representation of what is meant by quality—more detailed than criteria. The presence or absence of a particular metric in the software is a quantitative indication of the degree of quality present. This model is flexible in that it indicates a general relationship between categories. New factors, criteria, and metrics may be added without affecting the model.

2.1.2 Framework Elements

The initial framework contained eleven quality factors, twenty-three quality criteria, and thirty-nine metrics. Table 2.1-1 identifies and defines the quality factors. The factors

Table 2.1-1 Software Quality Factors

LIFE CYCLE STAGES	INITIAL PRODUCT OPERATION	CORRECTNESS	Extent to which a program satisfies its specification and fulfills the user's mission objectives.
		RELIABILITY	Extent to which a program can be expected to perform its intended function with required precision.
		EFFICIENCY	The amount of computing resources and code required by a program to perform a function.
		INTEGRITY	Extent to which access to software or data by unauthorized persons can be controlled.
		USABILITY	Effort required to learn, operate, prepare input, and interpret output of a program.
	PRODUCT REVISION	MAINTAINABILITY	Effort required to locate and fix an error in an operational program.
		TESTABILITY	Effort required to test a program to insure it performs its intended function.
		FLEXIBILITY	Effort required to modify an operational program.
	PRODUCT TRANSITION	PORTABILITY	Effort required to transfer a program from one hardware configuration and/or software system to another.
		REUSABILITY	Extent to which a program can be used in other applications - related to the packaging and scope of the functions that programs perform.
		INTEROPERABILITY	Effort required to couple one system with another.

NOTE: This table represents the original framework developed under contract F30602-78-C-0216 and was taken from RADC-TR-80-109, Volume I, Page 1-8.

are divided into three life cycle stages, or activity categories: initial product operation, product revision, and product transition. Each factor is most significant during a period when one of the three categories of activities is being performed.

Figure 2.1-1 (taken from RADC-TR-80-109, Volume I, pages 1-9 and 1-10) shows the initial twenty-three quality criteria in relationship to the eleven quality factors. Note that some criteria appear under more than one factor.

The initial thirty-nine metrics are identified in the next section (2.2) in Table 2.2-2 under "former metrics". Each metric is defined by a number of metric elements, with definitions for each metric element. This information was compiled into a set of metric tables, an example of which is shown in Table 2.1-2. The complete list of tables is contained in RADC-TR-80-109. The tables are used in conjunction with metric worksheets for tabulating metric values. The tables are categorized by criteria and subcriteria and indicate the applicable development phase(s) for each metric element.

2.2 FRAMEWORK ENHANCEMENTS

This section identifies the enhancements made to the baseline framework. New quality factors, criteria, metrics, and metric elements were identified; some baseline framework elements were modified. The software life-cycle phases were expanded from three to five. The worksheet categories were enhanced to accommodate the new and modified framework elements. The complete quality metrics framework for distributed systems is described in Section 3.0.

2.2.1 Quantitative Change Summary

Table 2.2-1 summarizes the quantitative changes to the baseline framework elements. The third column (New) indicates that even though only two quality factors were added, a considerable number of metric elements and worksheet references to those metric elements were added. This is because the basis of change to the framework elements stemmed from considering the effect of distributed systems on quality; this consideration affected all framework elements.

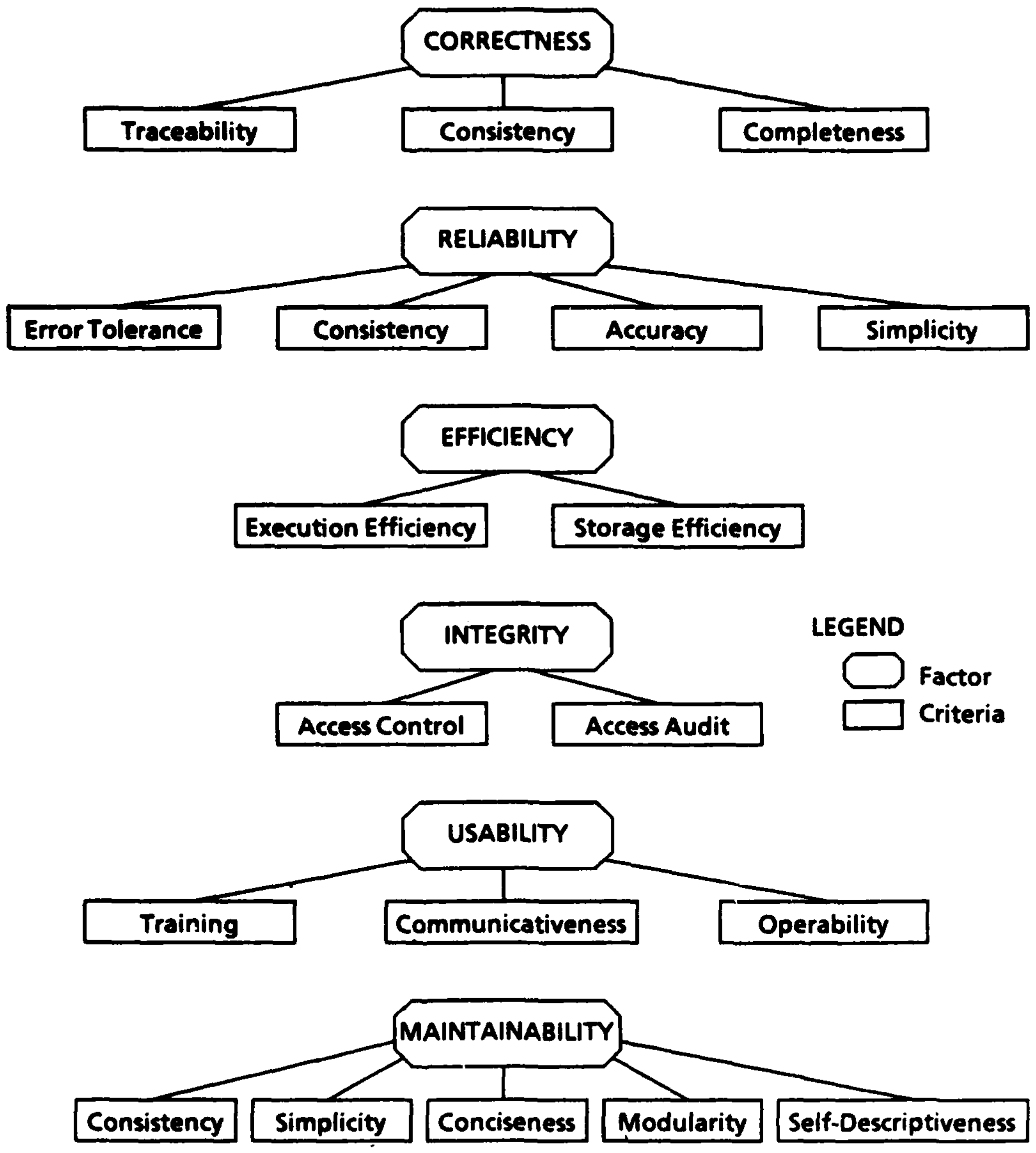


Figure 2.1-1 Relationship of Criteria to Software Quality Factors

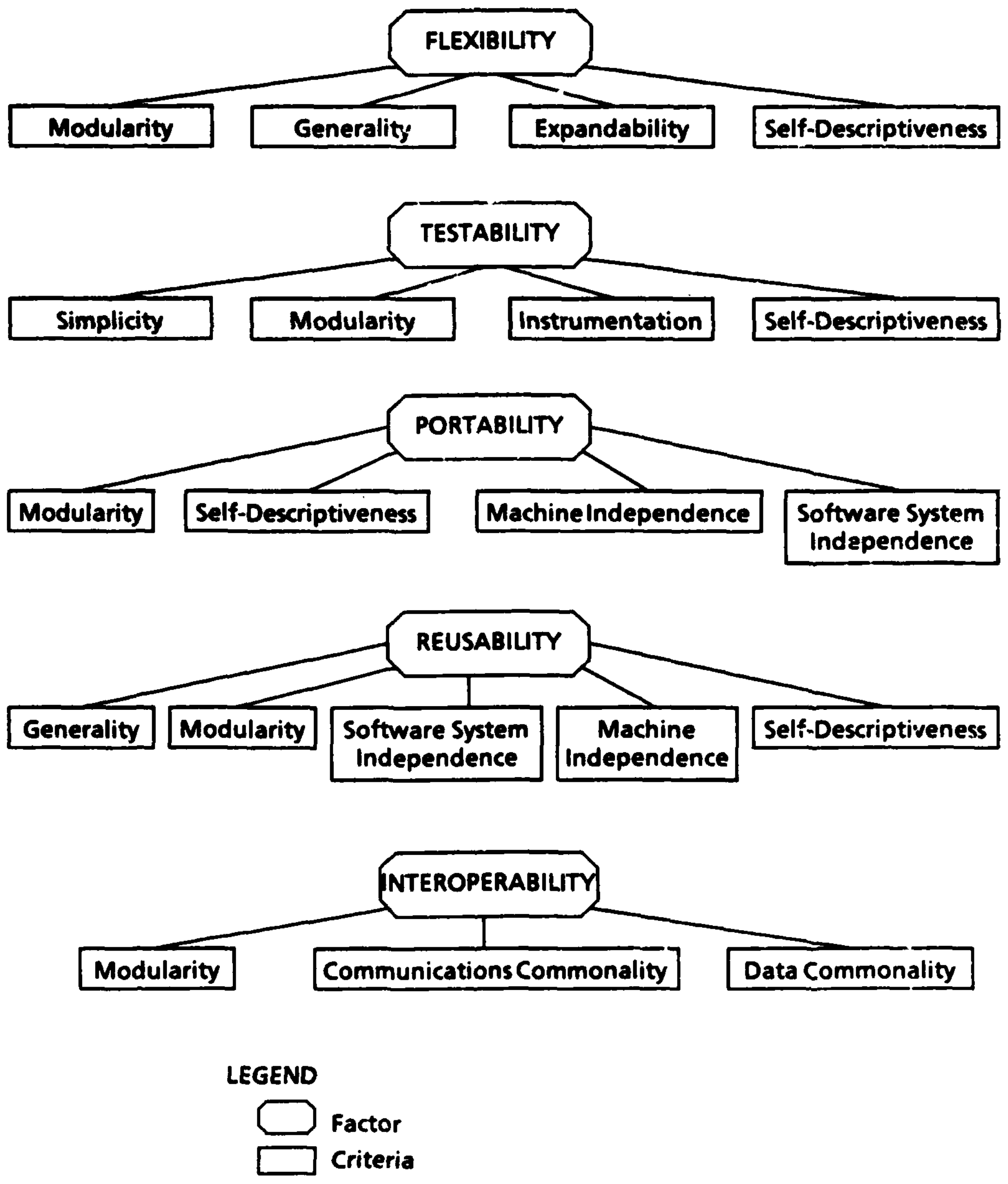


Figure 2.1-1 Relationship of Criteria to Software Quality Factors (Continued)

Table 2.1-2: Metric Table Example

FACTOR(S): RELIABILITY

CRITERION/ SUBCRITERION	METRIC	REQUIREMENTS		DESIGN		IMPLEMENTATION	
		YES/NO 1 OR 0	VALUE	YES/NO 1 OR 0	VALUE	YES/NO 1 OR 0	VALUE
INPUT DATA (p)	ET. 2 RECOVERY FROM IMPROPER INPUT DATA CHECKLIST:						
	(1) A definitive statement of requirement for error tolerance of input data.	1 II					
	(2) Range of values (reasonableness) for items specified and checked.			2b II		3 IV	
	(3) Conflicting requests and illegal combinations identified and checked.			2b II		3 IV	
	(4) All input is check before processing begins.			2b II		3 IV	
	(5) Determination that all data is available prior to procesing.			2b II		3 IV	
	METRIC VALUE: $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$						
RECOVERABLE COMPUTATIONAL FAILURES (p)	ET.3 · RECOVERY FROM COMPUTATIONAL FAILURES CHECKLIST:						
	(1) A definitive statement of requirement for recovery from computational faiures.	1 II					
	(2) Loop and multiple transfer index parameters range tested before use.			2b II		3 VII	
	(3) Subscript checking.			2b II		3 VII	
	(4) Critical output parameters reasonablesness checked during processing.			2b II		3 VII	
	METRIC VALUE: $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$						

Table 2.2-1 Metric Table And Worksheet Changes

	Former	Modified	New	Current	Proposed & Dropped
Metric Tables:					
Factors	11	1	2[+ 18%]	13	1
Criteria	23	11→7	5[+ 22%]	24	6
Metrics	39/2 *	0	11[+ 28%]	50/2 *	0
Metric Elements	154/26 *	9	67[+ 44%]	221/26 *	26
Worksheet References	202/3 *	6	119[+ 59%]	321/3 *	0
Metric Worksheets:					
Worksheet Categories	38	8	5[+ 13%]	43	2
Worksheet Entries	241/3 *	10	132[+ 55%]	373/3 * **	0

* = Active/Deleted

** = 45 of 373 are multiple entries

2.2.2 Metric Categorization

The baseline framework used both criteria and subcriteria in the categorization of the metrics. Table 2.2-2 shows the correlation between the baseline framework and the new framework. Subcriteria have been combined for simplicity. The criteria have also been alphabetized.

2.2.3 Life-Cycle Phases

The baseline life-cycle phases have been expanded from three to five. The baseline phases were requirements, design, and implementation. The new phases are requirements analysis, preliminary design, detailed design, implementation, and test and integration. Design was divided into preliminary design and detailed design. This enabled a direct correlation between worksheets and phases; each phase now has one worksheet. Worksheet 1 is used in requirements analysis; worksheet 2 is used in preliminary design; worksheet 3 is used in detailed design; worksheet 4 is used in implementation; and a subset of worksheet 2 is used in test and integration. The specific subset is indicated in the metric tables of Appendix B of Volume II of this report.

2.2.4 Worksheet Categories

The metric worksheet categories were revised as indicated in Table 2.2-3. New categories were added to accommodate new metrics. Some category names were revised and the category sequences were reordered to ease the task of data acquisition. Wherever possible, the same or similar metrics were grouped under the same category name and given the same category sub-number for each worksheet (e.g., structure - 1.1, 2.1, 3.1, and 4.1). There are five new categories indicated with an asterisk ("*") and eight category name changes indicated with a double asterisk ("**").

Note that Table 2.2-3 reflects the structure which was used in the validation performed under this contract. Several categories were added and changed when information from the Software Interoperability and Reusability contract F30602-80-C-0265 was integrated. These changes are reflected in Appendix A of Volume II of this report.

Table 2.2-2 Criteria and Metric Changes

REVISED		FORMER	
CRITERIA	METRIC	CRITERIA/ SUBCRITERIA	METRIC
ACCURACY	AY.1 Accuracy Checklist	ACCURACY	AC.1 Accuracy Checklist
ANOMALY MANAGEMENT	AM.1 Error Tolerance/Control Checklist	ERROR TOLERANCE/CONTROL	ET.1 Error Tolerance/Control Checklist
	AM.2 Improper Input Data Checklist	INPUT DATA	ET.2 Recovery From Improper Input Data Checklist
	AM.3 Computational Failures Checklist	RECOVERABLE COMPUTATIONAL FAILURES	ET.3 Recovery From Comuptational Failures Checklist
	AM.4 Hardware Faults Checklist	RECOVERABLE HARDWARE FAULTS	ET.4 Recovery From Hardware Faults Checklist
	AM.5 Device Errors Checklist	DEVICE STATUS CONDITIONS	ET.5 Recovery From Device Errors Checklist
AUGMENTABILITY	AG.1 Data Storage Expansion Measure	EXPANDABILITY/DATA STORAGE EXPANSION	EX.1 Data Storage Expansion Measure
	AG.2 Computation Extensibility Measure	COMPUTATION EXTENSIBILITY	EX.2 Extensibility Measure
COMMONALITY	CL.1 Communications Commonality Checklist	COMMUNICATIONS COMMONALITY	CC.1 Communications Commonality Checklist
	CL.2 Data Commonality Checklist	DATA COMMONALITY	DC.1 Data Commonality Checklist
COMMUNICATIVENESS	CM.1 User Input Interface Measure	COMMUNICATIVENESS/USER INPUT INTERFACE	CM.1 User Input Interface Measure
	CM.2 User Output Interface Measure	USER OUTPUT INTERFACE	CM.2 User Output Interface Measure
COMPLETENESS	CP.1 Completeness Checklist	COMPLETENESS	CP.1 Completeness Checklist
CONCISENESS	CO.1 Halstead's Measure	CONCISENESS	CO.1 Halstead's Measure
CONSISTENCY	CS.1 Procedure Consistency Measure	CONSISTENCY/PROCEDURE CONSISTENCY	CS.1 Procedure Consistency Measure
	CS.2 Data Consistency Measure	DATA CONSISTENCY	CS.2 Data Consistency Meausre
EFFECTIVENESS	EF.1 Performance Requirements	EXECUTION EFFICIENCY/REQUIREMENTS	EE.1 Performance Requirements Identified and allocated to Design
	EF.2 Iterative Processing Efficiency Measure	ITERATIVE PROCESSING	EE.2 Iterative Processing Efficiency Measure
	EF.3 Data Usage Efficiency Measure	DATA USAGE	EE.3 Data Usage Efficiency Measure
	EF.4 Storage Efficiency Measure	STORAGE EFFICIENCY	SE.1 Storage Efficiency Measure
GENERALITY	GE.1 Module References by Other Modules	GENERALITY REFERENCES	GE.1 Extent to Which Module is Referenced by Other Modules
	GE.2 Implementation for Generality Checklist	IMPLEMENTATION GENERALITY	GE.2 Implementation for Generality Checklist
INDEPENDENCE	ID.1 Software System Independence Measure	SOFTWARE SYSTEM INDEPENDENCE	SS.1 Software System Independence Measure
	ID.2 Machine Independence Measure	MACHINE INDEPENDENCE	MI.1 Machine Independence Measure

Table 2.2-2 Criteria and Metric Changes (Continued)

REVISED		FORMER	
CRITERIA	METRIC	CRITERIA/ SUBCRITERIA	METRIC
MODULARITY	DELETED	MODULARITY/DEGREE OF INDEPENDENCE	MO.1 Stability Measure
	MO.2 Modular Implementation Measure	MODULAR IMPLEMENTATION	MO.2 Modular Implementation Measure
OPERABILITY	OP.1 Operability Checklist	OPERABILITY	OP.1 Operability Checklist
SELF-DESCRIPTIVENESS	SD.1 Quantity of Comments	SELF-DESCRIPTIVENESS/ QUANTITY OF COMMENTS	SD.1 Quantity of Comments
	SD.2 Effectiveness of Comments Measure	EFFECTIVENESS OF COMMENTS	SD.2 Effectiveness of Comments Measure
	SD.3 Descriptiveness of Language Measure	DESCRIPTIVENESS OF IMPLEMENTATION LANGUAGE	SD.3 Descriptiveness of Implementation Language Measure
SIMPLICITY	SI.1 Design Structure Measure	SIMPLICITY/DESIGN STRUCTURE	SI.1 Design Structure Measure
	SI.2 Structured Language or Preprocessor	STRUCTURE PROGRAMMING	SI.2 Use of Structure Language or Preprocessor
	SI.3 Data and Control Flow Complexity Measure	DATA AND CONTROL FLOW COMPLEXITY	SI.3 Complexity Measure
	SI.4 Coding Simplicity Measure	CODE SIMPLICITY	SI.4 Measure of Coding Simplicity
SYSTEM ACCESSIBILITY	SA.1 Access Control Checklist	ACCESS CONTROL	AC.1 Access Control Checklist
	SA.2 Access Audit Checklist	ACCESS AUDIT	AA.1 Access Audit Checklist
TRACEABILITY	TR.1 Cross Reference	TRACEABILITY	TR.1 Cross Reference Relating Modules to Requirement
TRAINING	TN.1 Training Checklist	TRAINING	TN.1 Training Checklist
VISIBILITY	VS.1 Module Testing Measure	INSTRUMENTATION/MODULE TESTING SUPPORT	IN.1 Module Testing Measure
	VS.2 Integration Testing Measure	INTEGRATION TESTING SUPPORT	IN.2 Integration Testing Measure
	VS.3 System Testing Measure	SYSTEM TESTING SUPPORT	IN.3 System Testing Measure

Table 2.2-3 Metric Worksheets Summary

1 REQUIREMENTS ANALYSIS/SYSTEM LEVEL	2 PRELIMINARY DESIGN/SYSTEM LEVEL	3 DETAILED DESIGN/MODULE LEVEL	4 SOURCE CODE/MODULE LEVEL
(*) 1.1 Structure * {RL, SV, MA, VE, FX, PO, RU, IP, EX} SI.1, DI.1* SI.4, MO.3*	(III) 2.1 Structure {RL, IG, US, SV, MA, VE, FX, PO, RU, IP, EX} SI.1, DI.1*, VR.1*, MO.3*	(III) 3.1 Structure {CO, RL, SV, MA, VE, FX, PO, RU, IP, EX} SI.1, SI.3, SI.4, DI.1*, SP.1*, MO.3*	(I) 4.1 Structure {RL, SV, MA, VE, FX, PO, RU, IP, EX} MO.2, SI.1, SI.2, SI.3, SI.4
(II) 1.2 Tolerance** {RL, SV} AY.1, AM.2, AM.3, AM.4, AM.5, AM.6*, AM.7*	(II) 2.2 Tolerance** {RL, SV} AY.1, AM.1, AM.4, AM.5, AM.6* AM.7*	(II) 3.2 Tolerance** {RL, SV} AY.1, AM.1, AM.2, AM.3, AM.6*, AM.7*	(VII) 4.2 Tolerance** {RL, SV} AM.1, AM.3
(V) 1.3 Performance {EF} EF.1	(IV) 2.3 Optimization {EF} EF.2, EF.3, EF.4	(VI) 3.3 Optimization {EF} EF.1, EF.2, EF.3	(VIII) 4.3 Optimization** {EF} EF.2, EF.3, EF.4
(I) 1.4 Completeness {CO} CP.1, TR.1	(I) 2.4 Completeness {CO} TR.1, CP.1	(I) 3.4 Completeness {CO} CP.1, TR.1*	(II) 4.4 Conciseness {MA} CO.1
(III) 1.5 Security {IG} SA.1, SA.2	(V) 2.5 Security {IG} SA.1, SA.2	(IV) 3.5 References {SV, MA, VE, FX, PO, RU, IP, EX} ID.1, ID.2, MO.2	(V) 4.5 References {SV, MA, VE, FX, PO, RU, IP, EX} MO.2, ID.1
(*) 1.6 Changeability* {FX, EX} AG.1, AG.2, AG.3*, AG.4*	(*) 2.6 Changeability* {FX, RU, EX} AG.1, AG.2, AG.3*, AG.4*, GE.1	(V) 3.6 Changeability** {FX, RU, EX} AG.1, AG.2, AG.3*, GE.2	(X) 4.6 Changeability** {FX, RU, EX} AG.1, AG.2, GE.2
(VI) 1.7 System Interfaces {SV, IP} CL.1, CL.2, AU.1*, AU.2*, RE.1*	(VI) 2.7 System Interfaces {SV, IP} CL.1, CL.2, AU.1*, AU.2*, RE.1*	(*) 3.7 System Interfaces* {SV} AU.1*	(IV) 4.7 Input/Output {RL, SV, PO, RU} ID.2, AM.2, AU.1*
(*) 1.8 Data Base *{CO, RL, IG, US, SV, MA, EX} RE.1*, DI.1*, VR.1*, CS.2*	(IX) 2.8 Data Base {CO, RL, IG, US, SV, MA, VE, FX, RU, EX} SI.1, RE.1*, DI.1*, VR.1*, CS.2*	(VIII) 3.8 Consistency {CO, RL, MA} CS.1, CS.2	(III) 4.8 Self-Descriptiveness {MA, VE, FX, PO, RU} SD.1, SD.2, SD.3
(IV) 1.9 Human Interface {US} OP.1, CM.1, CM.2	(VII) 2.9 Human Interface {US} OP.1, TN.1, CM.1, CM.2	(VII) 3.9 Functional Categorization	(VI) 4.9 Data {CO, RL, EF, MA, VE, FX, RU, EX} SI.4, EF.4, CS.2
(VII) 1.10 Inspector's Comments	(VIII) 2.10 Testing {US, MA, VE} VS.1, VS.2, VS.3	(IX) 3.10 Inspector's Comments	(IX) 4.10 Independence** {PO, RU} ID.1, ID.2
	(X) 2.11 Inspector's Comments		(XI) 4.11 Dynamic Measurement {RL, EF, SV, FX, EX} AY.1, AG.2*, AG.3*, EF.3, EF.4, AU.1*
			(XII) 4.12 Inspector's Comments

* = New, ** = Modified, () = Former Numbering Scheme

The acronyms used in Table 2.2-3 for the quality factors are as follows:

CO	- Correctness
RL	- Reliability
EF	- Efficiency
IG	- Integrity
US	- Usability
SV	- Survivability
MA	- Maintainability
VE	- Verificability
FX	- Flexibility
PO	- Portability
RU	- Reusability
IP	- Interoperability
EX	- Expandability

2.3 TRANSITION TO NEW FRAMEWORK: SUMMARY OF VOLUME III

Volume III of this report is titled "Distributed Computing Systems: Impact on Software Quality". What follows is a summary of the contents of that volume, presented in the context of the transition from the old quality metrics framework to the new framework.

Distributed Computer Systems have been variously defined, and these definitions are compared. An historical overview is provided, as well as the relationship to the current DoD software initiative.

Over 50 rationales are given for the selection of a distributed system rather than a uniprocessor system. These rationales are grouped into 9 reasons, where each reason is a high-level system acquisition goal. A matrix relates these rationales and reasons to those quality factors most impacted. Within each reason there are tradeoffs, and tables illustrate the tradeoffs.

There are three areas which must be addressed in the design of a distributed system. These areas are: the distribution of control and processing, the structure and distribution of the data base, and the strategy for communication among the elements of the system.

Successful design strategies in each area are outlined, with an emphasis on quality factors.

Distributed system topology is a term referring to the physical or logical pattern of interconnection of system components. Aspects of distributed system topology discussed in Volume III include:

* topology impact - how and why topology is related to system quality factors
* communications strategies - differing approaches to routing messages between nodes
* distributed system layers - the ISO Reference Model for interprocessor communication protocols
* distributed system architecture classification - what topologically different designs are possible, and their relative advantages
* distributed system hardware architecture (topology) impact on system quality
* distributed system hardware (topology) impact on software quality

Fourteen particular topologies are illustrated, from the familiar (loop, ring, bus) to the more exotic (cube-connected cycles, binary hypercube, shuffle-exchange).

A number of scenarios were developed that collectively cover many of the major system and software quality allocation issues that arise in distributed systems. Scenarios include: distributed command and control, distributed communications, distributed database, distributed avionics, distributed functional testing, distributed space systems, distributed virtual topology, distributed optoelectronics, and distributed microcircuit multiprocessor.

A tentative classification is introduced for the decisions available to software, hardware, and system acquisition managers. The concept of a distributed system life cycle is outlined.

A number of firmware issues are explored. These include the difficulties of procurement, an example of Ada implementation, and the quality factors involved in hardware/firmware/software tradeoffs. A classification is introduced which emphasizes nardware/firmware/software tradeoff opportunities in memory management, operational management, I/O management, error monitoring, and special algorithmic capabilities.

Efficiency, virtuality, adaptation, performance, and reliability are discussed in terms of direct and indirect effects of hardware architecture on system quality. Quality factor emphasis and methodology as a whole are discussed, as well as qualitative relationships between correctness and reliability.

Collectively, these qualitative studies of distributed computing systems and their impacts on quality raised a number of issues which were not considered in the previous (uniprocessor) framework. New factors, criteria, and metrics were developed in order to quantitatively assess these issues. There is not a one-to-one mapping between the discussions and scenarios in Volume III and the new factors, criteria, and metrics. Instead, the material in Volume III illustrates the learning process of researchers in this contract which enabled the quality metrics framework to be extended to the more complex domain of distributed computing systems.

3.0
DISTRIBUTED SYSTEM QUALITY METRICS FRAMEWORK

The goal of the quality metrics concept is to enable a software acquisition manager to specify the types and degrees of software qualities desired in the end product and to predict and measure the degree of presence of those qualities during the development process. Previous work (see Section 2.0) has established a model for viewing software quality. Figure 3.0-1 is a simple depiction of this model, showing an hierarchical relationship between a quality factor, quality criteria, and quality metrics. Quality factors (e.g., usability, correctness, maintainability) are user-oriented terms representing concerns of the acquisition manager or product user. Quality factors are used to specify the type of quality desired. Criteria are software-oriented terms representing attributes of the software which, if present in the software, indicate the presence of a type of quality (a quality factor). Operability, communicativeness, and visibility are criteria for the quality factor usability. Metrics are software-oriented phrases or sentences which ask questions concerning details of an attribute (criterion) of the software. Answers to the questions enable quantification of the degree of presence of criteria and, hence, factors. "All error conditions and responses appropriately described to operator" is an example from the metric checklist for the criteria operability.

The desired quality factors are normally specified by the acquisition manager and provided as part of the requirements (along with operation, performance, and design requirements). This enables the corresponding criteria and metrics to be identified and used to predict the degree of presence of the desired qualities at key review points during the development process.

Part of the purpose of this contract was to determine suitable quality metrics that will directly apply to software on distributed systems. Previous work has established factors, criteria, and metrics applicable to uniprocessor systems. An investigation of user concerns and software characteristics for distributed systems shows that the previously established factors and criteria are equally applicable to distributed systems. The investigation also resulted in identification of new quality factors, criteria, and metrics that are unique to distributed systems. The following paragraphs describe the full set of quality factors, criteria, and metrics; new factors, criteria, and metrics are identified with an asterisk. A complete description of the metrics framework and the use of the metrics technology appears in the handbook—Volume II of this report.

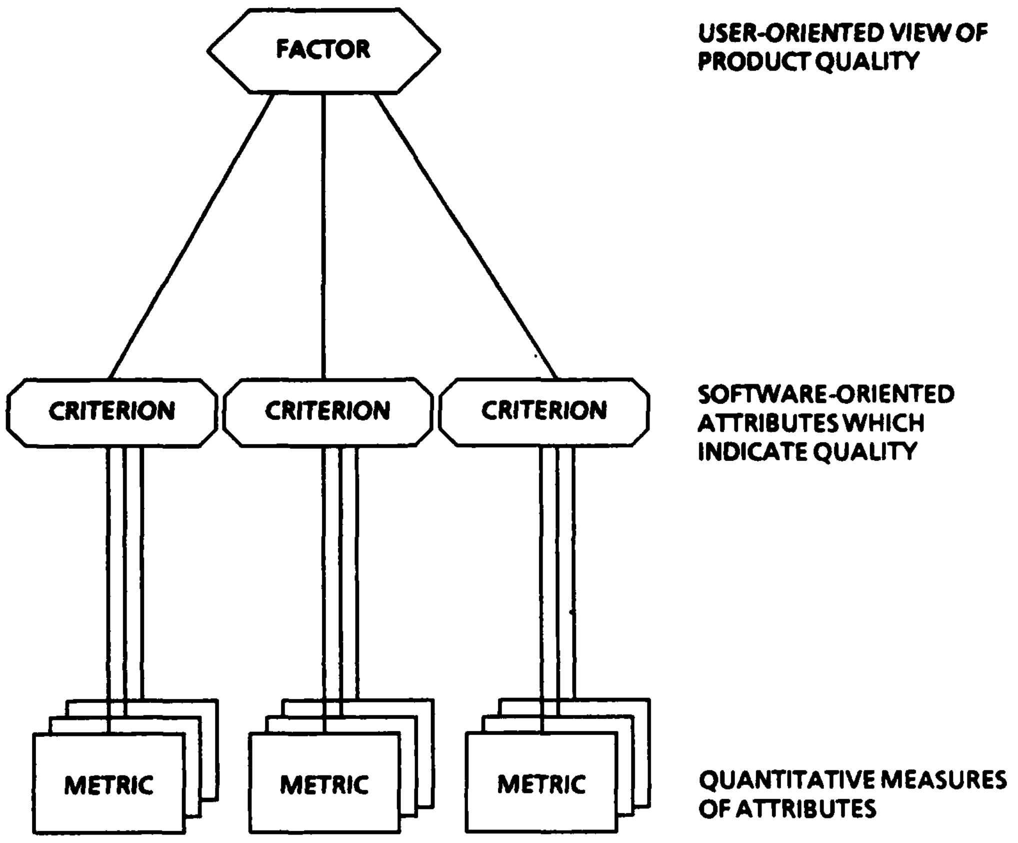

FACTOR
USER-ORIENTED VIEW OF PRODUCT QUALITY
CRITERION
CRITERION
CRITERION
SOFTWARE-ORIENTED ATTRIBUTES WHICH INDICATE QUALITY
METRIC
METRIC
METRIC
QUANTITATIVE MEASURES OF ATTRIBUTES

Figure 3.0-1 Software Quality Model

3.1 SOFTWARE QUALITY FRAMEWORK

This section describes the distributed systems software quality framework, including factors, criteria, metrics, tables, and worksheets. Two new factors, five new criteria, and eleven new metrics were added to the software quality framework.

3.1.1 Software Quality Factors

Previous analysis and evaluation of quality factors resulted in grouping quality factors into three categories: product operation, product revision, and product transition. This scheme emphasizes the user's view of software life cycle management, and, in addition, it includes conversion or reapplication of the software. Table 3.1-1 illustrates this scheme. The questions indicate the relevancy of the factors to a user. Table 3.1-2 provides definitions for all of the quality factors. Two new quality factors were identified and one factor was revised as discussed below:

Product Operation

The quality factor survivability was added to the category of product operation. Because components in a distributed system are physically separated, it is not uncommon for large portions of a system to remain operative when a single unit fails. For command, control and communications applications, users are concerned that critical functions continue to be supported by the system even when a portion of the system is inoperable.

Product Revision

The quality factor testability was changed to verifiability, a more general term. Testing is one method of verification. Other methods include analysis, inspection, and demonstration. In systems where reliability is critical, testing is often augmented by analysis and inspection.

Product Transition

The quality factor expandability was added to the category of product transition.

Table 3.1-1 Quality Life-Cycle Scheme

Activity	User Concern	Quality Factor
PRODUCT OPERATION	DOES IT DO WHAT IT'S SUPPOSED TO?	CORRECTNESS
	WHAT CONFIDENCE CAN BE PLACED IN WHAT IT DOES?	RELIABILITY
	HOW WELL DOES IT UTILIZE THE RESOURCES?	EFFICIENCY
	HOW SECURE IS IT?	INTEGRITY
	HOW EASY IS IT TO USE?	USABILITY
	HOW WELL WILL IT PERFORM UNDER ADVERSE CONDITIONS?	SURVIVABILITY*
PRODUCT REVISION	CAN IT BE REPAIRED?	MAINTAINABILITY
	CAN ITS OPERATION AND PERFORMANCE BE VERIFIED?	VERIFIABILITY*
	CAN IT BE CHANGED?	FLEXIBILITY
PRODUCT TRANSITION	CAN IT BE USED IN ANOTHER ENVIRONMENT?	PORTABILITY
	CAN IT BE USED IN ANOTHER APPLICATION?	REUSABILITY
	CAN IT BE INTERFACED WITH ANOTHER SYSTEM?	INTEROPERABILITY
	CAN ITS CAPABILITY OR PERFORMANCE BE EXPANDED OR UPGRADED?	EXPANDABILITY*

* = NEW OR MODIFIED

Table 3.1-2 Software Quality Factor Definitions

Activity	Quality Factor	Definition
OPERATION	CORRECTNESS	EXTENT TO WHICH THE SOFTWARE SATISFIES ITS SPECIFICATIONS AND FULFILLS THE USER'S MISSION OBJECTIVES
	RELIABILITY	PROBABILITY THAT THE SOFTWARE WILL PERFORM ITS LOGICAL OPERATIONS IN THE SPECIFIED ENVIRONMENT WITHOUT FAILURE
	EFFICIENCY	DEGREE OF UTILIZATION OF RESOURCES (PROCESSING TIME, STORAGE, COMMUNICATION TIME) IN PERFORMING FUNCTIONS
	INTEGRITY	EXTENT TO WHICH UNAUTHORIZED ACCESS TO THE SOFTWARE OR DATA CAN BE CONTROLLED
	USABILITY	EFFORT FOR TRAINING AND SOFTWARE OPERATION - FAMILIARIZATION, INPUT PREPARATION, EXECUTION, OUTPUT INTERPRETATION
	SURVIVABILITY*	PROBABILITY THAT THE SOFTWARE WILL CONTINUE TO PERFORM OR SUPPORT CRITICAL FUNCTIONS WHEN A PORTION OF THE SYSTEM IS INOPERABLE
REVISION	MAINTAINABILITY	AVERAGE EFFORT TO LOCATE AND FIX A SOFTWARE FAILURE
	VERIFIABILITY*	EFFORT TO VERIFY THE SPECIFIED SOFTWARE OPERATION AND PERFORMANCE
	FLEXIBILITY	EFFORT TO EXTEND THE SOFTWARE MISSIONS, FUNCTIONS, OR DATA TO SATISFY OTHER REQUIREMENTS
TRANSITION	PORTABILITY	EFFORT TO CONVERT THE SOFTWARE FOR USE IN ANOTHER OPERATING ENVIRONMENT (HARDWARE CONFIGURATION, SOFTWARE SYSTEM ENVIRONMENT)
	REUSABILITY	EFFORT TO CONVERT A SOFTWARE COMPONENT FOR USE IN ANOTHER APPLICATION
	INTEROPERABILITY	EFFORT TO COUPLE THE SOFTWARE OF ONE SYSTEM TO THE SOFTWARE OR ANOTHER SYSTEM
	EXPANDABILITY*	EFFORT TO INCREASE SOFTWARE CAPABILITY OR PERFORMANCE BY ENHANCING CURRENT FUNCTIONS OR ADDING NEW FUNCTIONS/DATA

* = NEW OR MODIFIED

Distributed systems often evolve in size and capability over time, and older nodes and components often require an upgrade in capability and/or performance. This quality factor addresses the user's concern for the high cost of developing new systems through emphasizing the extension of the life cycle of a system.

3.1.1.1 Relationship of Quality Factors to Life-Cycle Phases

Software quality affects all phases of the software life cycle. Quality factor requirements are specified during the concept formulation phase, and quality ratings are estimated and predicted during the development phases. The effect of the presence or absence of a particular quality factor is realized during the evaluation phase and during the post-development phases after the product has been turned over to the user. Table 3.1-3 indicates the development phases where quality levels are determined for each quality factor and indicates the evaluation and post-development phases where the effect of each quality factor is realized. Previous work had identified three development phases where quality levels are determined. This has been expanded to five development phases: Requirements Analysis, Preliminary Design, Detailed Design, Implementation, and Test and Integration.

The degree of impact of poor quality determines the cost savings that can be expected if a higher quality product is developed through the application of metrics. This potential cost savings is reduced by the additional costs to apply the metrics and to develop the higher quality product. The expected-cost-saved/cost-to-provide ratio for each quality factor is rated as high, medium, or low in the right hand column of Table 3.1-3.

3.1.1.2 Relationships Between Software Quality Factors

Specifying more than one type of quality for a product can affect cost in a nonlinear fashion. Relationships exist between quality factors--some are complementary while others conflict. The impact of conflicting factors is that the cost-to-provide increases. Table 3.1-4 shows the effect of a high degree of quality for one factor upon each of the other factors. For example, reliability and correctness are complementary. If a high degree of reliability is present, a high degree of correctness would also be expected; and vice versa. Efficiency and reliability conflict. If a high degree of efficiency is present, a low degree of reliability would be expected; and vice versa. Table 3.1-5 is a summary of

Table 3.1-3 Relationship of Quality Factors to Life-Cycle Phases

	Development					Eval	Post-Development			Benefit
Life-Cycle Phases / Factors	Reqmts Analysis	Prelim Design	Detail Design	Imple-mentation	Test & Integr	System Test & Instl	Operation	Revision	Transition	Expected Cost-Saved/Cost-to-Provide Ratio
Correctness	△	△	△	△	△	X	X	X		High
Reliability	△	△	△	△	△	X	X	X		High
Efficiency	△	△	△	△	△		X			Low
Integrity	△	△			△		X			Low
Usability	△	△			△	X	X	X		Medium
Survivability*	△	△	△	△	△	X	X			Low
Maintainabilty	△	△	△	△	△			X	X	High
Verifiability*	△	△	△	△	△	X		X	X	High
Flexibility	△	△	△	△	△			X	X	Medium
Portability	△	△	△	△					X	Medium
Reusability	△	△	△	△	△				X	Medium
Interoperability	△	△	△	△	△	X			X	Medium
Expandability*	△	△	△	△	△				X	Medium

Legend: △ = When Quality is Measured and Predicted, X = When Impact of Poor Quality is Recognized

* New or Modified

Table 3.1-4 Relationships Between Software Quality Factors

LEGEND:
IF A HIGH DEGREE OF QUALITY
IS PRESENT FOR ONE FACTOR,
THE DEGREE OF QUALITY EXPECTED
FOR THE OTHER FACTOR IS:

△ = HIGH
▲ = LOW

BLANK = NONE OR DEPENDENT UPON APPLICATION

* = NEW OR MODIFIED

SOFTWARE QUALITY FACTORS / SOFTWARE QUALITY FACTORS	CORRECTNESS	RELIABILITY	EFFICIENCY	INTEGRITY	USABILITY	SURVIVABILITY*	MAINTAINABILITY	VERIFIABILITY*	FLEXIBILITY	PORTABILITY	REUSABILITY	INTEROPERABILITY	EXPANDABILITY*
CORRECTNESS													
RELIABILITY	△												
EFFICIENCY		▲											
INTEGRITY			▲										
USABILITY	△	△	▲										
SURVIVABILITY*	△	△	▲	▲	△								
MAINTAINABILITY	△	△	▲		△								
VERIFIABILITY*	△	△	▲	△	△		△						
FLEXIBILITY	△	▲	▲	▲	△	▲	△	△					
PORTABILITY			▲			▲			△				
REUSABILITY	△	▲	▲	▲	△	▲	△	△	△	△			
INTEROPERABILITY		△	▲	▲					△		△		
EXPANDABILITY*	△	▲	▲	▲	△		△	△	△	△	△	△	

Table 3.1-5 Relationships Between Software Use and Quality Factors

	USAGE CATEGORY												
	OPERATION						REVISION			TRANSITION			
QUALITY FACTORS / USAGE CATEGORY	CORRECTNESS	RELIABILITY	EFFICIENCY	INTEGRITY	USABILITY	SURVIVABILITY*	MAINTAINABILITY	VERIFIABILITY*	FLEXIBILITY	PORTABILITY	REUSABILITY	INTEROPERABILITY	EXPANDABILITY*
OPERATION	△	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
REVISION	△	▲	▲	▲	△	▲	△	△	△	△	△	△	△
TRANSITION	△	▲	▲	▲	△	▲	△	△	△	△	△	△	△

LEGEND:

IF A HIGH DEGREE OF QUALITY IS PRESENT FOR ONE FACTOR, THE DEGREE OF QUALITY EXPECTED FOR OTHER FACTORS OF A USAGE CATEGORY IS:

△ = HIGH
▲ = LOW

* = NEW OR DF MODIFIED

Table 3.1-4 and indicates the effect of each quality factor on the usage categories (operation, revision, and transition). In general, the operation quality factors conflict among themselves and conflict with the revision and transition quality factors; the revision and transition factors do not conflict among themselves and are usually synergistic. Table 3.1-6 provides a brief discussion of typical tradeoffs for conflicting factors. The first entry explains the conflict between reliability and efficiency which was noted in the example above for Table 3.1-4.

3.1.1.3 Software Quality Ratings

Quality ratings have been developed for eight of the thirteen quality factors. Table 3.1-7 shows the formulas which have been developed and an explanation of the terms. These ratings are used in conjunction with a set of guidelines for specifying the quality level required for a quality factor. Two new ratings were developed under this contract—the ratings for survivability and expandability.

3.1.2 Software Quality Criteria

Previous analysis and evaluation had resulted in identification of twenty three quality criteria. The investigation of software characteristics for distributed systems showed that these criteria were also applicable to distributed systems. This investigation resulted in the identification of a total of twenty-four quality criteria. Twelve criteria are new; eleven of the previous criteria were revised and combined into seven criteria as discussed below.

Figure 3.1-1 shows the criteria which are associated with each of the thirteen quality factors. This figure is in the format of the software quality model which was depicted in Figure 3.0-1. Table 3.1-8 also shows the criteria associated with each of the thirteen quality factors; in addition, it indicates which relationships had been established by previous analysis and evaluation (with an "x"), which relationships are new (with an "$\boxed{x}$"), and which criteria and factors are new (with an "*") or different (with an "**"). Table 3.1-9 provides definitions for each of the quality criteria and lists the related factors. The criteria in the tables are listed in alphabetical order.

Table 3.1-6 Typical Factor Tradeoffs

RELIABILITY VS	EFFICIENCY	THE ADDITIONAL CODE REQUIRED TO PROVIDE ACCURACY AND TO PERFORM ANOMALY MANAGEMENT USUALLY INCREASES RUN TIME AND REQUIRES ADDITIONAL STORAGE.
	FLEXIBILITY REUSABILITY EXPANDABILITY	THE GENERALITY REQUIRED FOR FLEXIBLE, REUSABLE, AND EXPANDABLE SOFTWARE USUALLY INCREASES THE DIFFICULTY OF PROVIDING ACCURACY AND PERFORMING ANOMALY MANAGEMENT FOR SPECIFIC CASES.
EFFICIENCY VS	INTEGRITY	THE ADDITIONAL CODE AND PROCESSING REQUIRED TO CONTROL ACCESS TO CODE OR DATA USUALLY LENGTHENS RUN TIME AND REQUIRES ADDITIONAL STORAGE.
	USABILITY	THE ADDITIONAL CODE AND PROCESSING REQUIRED TO EASE AN OPERATOR'S TASK OR TO PROVIDE MORE USABLE OUTPUT USUALLY INCREASE RUNTIME AND REQUIRE ADDITIONAL STORAGE.
	SURVIVABILITY	THE ADDITIONAL CODE AND PROCESSING REQUIRED FOR MODULAR, RECONFIGURABLE, ANOMALY TOLERANT SOFTWARE RESULTS IN LESS EFFICIENT OPERATION.
	MAINTAINABILITY VERIFIABILITY	USING MODULAR, VISIBLE, SELF-DESCRIPTIVE CODE TO INCREASE MAINTAINABILITY AND VERIFIABILITY USUALLY INCREASES OVERHEAD AND RESULTS IN LESS EFFICIENT OPERATION. CODE WHICH IS OPTIMIZED FOR EFFICIENCY POSES PROBLEMS TO THE TESTER & MAINTAINER.
	FLEXIBILITY REUSABILITY	THE GENERALITY REQUIRED FOR FLEXIBLE AND REUSABLE SOFTWARE INCREASES OVERHEAD AND DECREASES EFFICIENCY.
	PORTABILITY	THE USE OF CODE OPTIMIZED FOR EFFICIENCY USUALLY DECREASES PORTABILITY.
	INTEROPERABILITY	THE OVERHEAD FOR CONVERSION FROM STANDARD DATA REPRESENTATIONS AND FOR THE USE OF STANDARD INTERFACE ROUTINES DECREASES OPERATING EFFICIENCY.
	EXPANDABILITY	THE USE OF MODULAR, GENERAL SOFTWARE USUALLY DECREASES OPERATING EFFICIENCY.
INTEGRITY VS	SURVIVABILITY	THE DISTRIBUTEDNESS REQUIRED FOR SURVIVABLE SOFTWARE INCREASES THE RISK OF UNAUTHORIZED ACCESS.
	FLEXIBILITY REUSABILITY	THE GENERALITY REQUIRED FOR FLEXIBLE AND REUSABLE SOFTWARE INCREASES THE RISK OF UNAUTHORIZED ACCESS.
	INTEROPERABILITY	COUPLED SYSTEM HAVE MORE AVENUES OF ACCESS, DIFFERENT USERS, AND COMMON DATA REPRESENTATIONS; THE Y OFTEN SHARE DATA AND CODE. THESE INCREASE THE POTENTIAL FOR ACCIDENTAL OR DELIBERATE ACCESS OF SENSITIVE DATA.
	EXPANDABILITY	THE GENERALITY REQUIRED FOR EXPANDABLE SOFTWARE INCREASES THE RISK OF UNAUTHORIZED ACCESS.
SURVIVABILITY VS	FLEXIBILITY PORTABILITY REUSABILITY	THE RECONFIGURABILITY REQUIRED FOR SURVIVABLE SOFTWARE REDUCES ITS FLEXIBILITY, PORTABILITY, AND REUSABILITY

Table 3.1-7 Software Quality Ratings

Quality Factor	Rating	Explanation
Correctness	**	
Reliability	$1 - \frac{\text{Number of Errors}}{\text{Number of Lines Of Code}}$	In Terms Of The Number Of Errors That Occur After The Start Of Formal Testing
Efficiency	**	
Integrity	**	
Usability	**	
Survivability*	$1 - \frac{\text{Number of Errors}}{\text{Words Of Object Code}}$	In Terms Of The Number Of Survivability Related Errors That Occur After the Start of Formal Testing
Maintainability	1 − 0.1 (Average Number of Man-Days To Fix)	In Terms Of Average Effort To Locate and Fix An Error
Verifiability*	**	
Flexibility	1 − 0.05 (Average Number of Man-Days To Change)	In Terms Of Average Effort To Extend Software To Satisfy Other Requirements
Portability	$1 - \frac{\text{Effort To Transport}}{\text{Effort to Implement}}$	In Terms Of Effort To Convert Software For Use In Another Environment And The Original Implementation Effort
Reusability	$1 - \frac{\text{Effort To Convert}}{\text{Effort To Develop}}$	In Terms Of Effort To Convert Software For Use In Another Application And The Original Development Effort
Interoperability	$1 - \frac{\text{Effort To Couple}}{\text{Effort To Couple} + \text{Effort to Develop}}$	In Terms Of Effort To Couple Software And The Original Development Effort
Expandability*	$1 - \frac{\text{Effort To Expand}}{\text{Effort To Develop}}$	In Terms Of Effort To Increase Software Capability/Performand and Original Development Effort

* = New or Modified
** = None Development

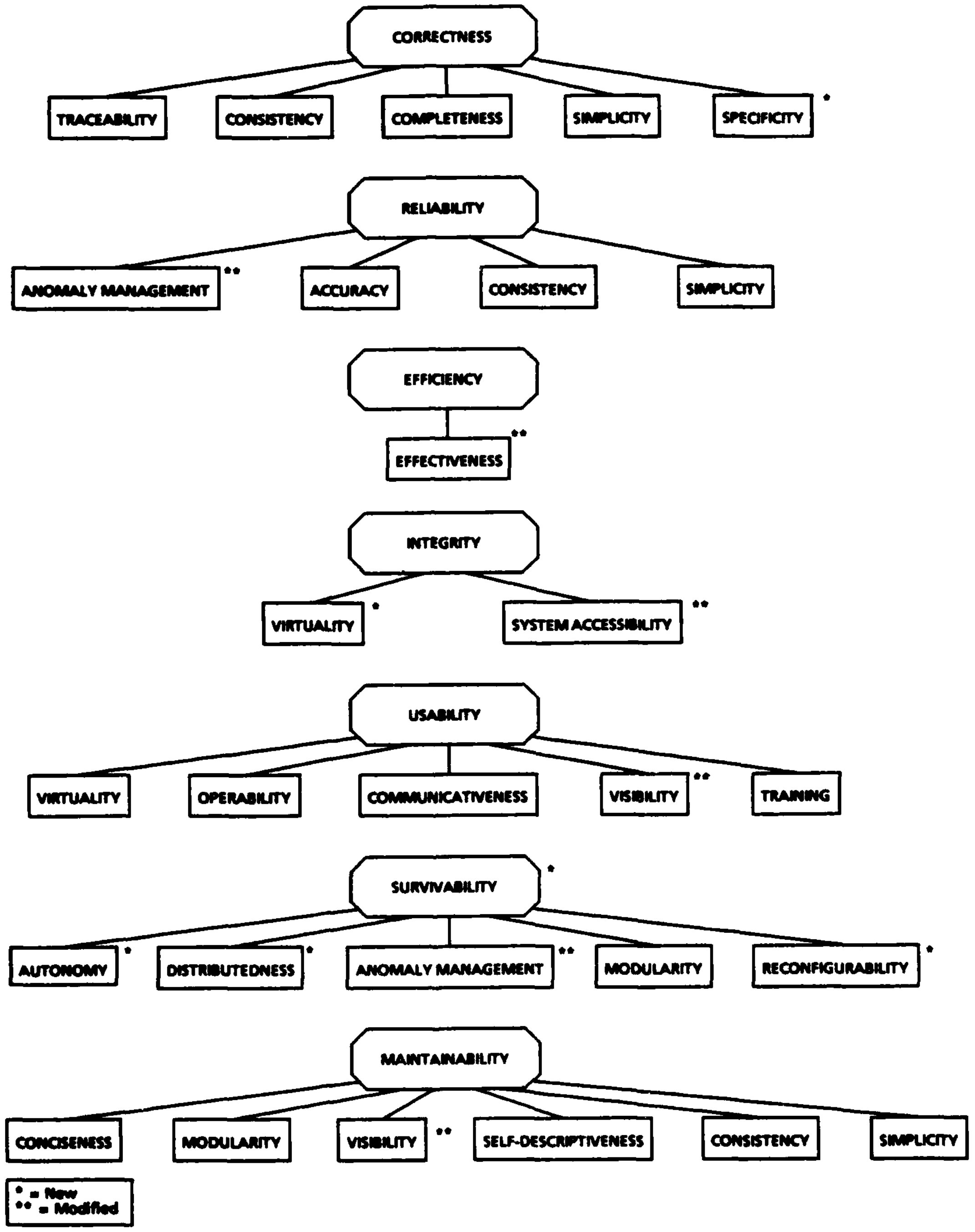


Figure 3.1-1 Relationship of Criteria to Software Quality Factors

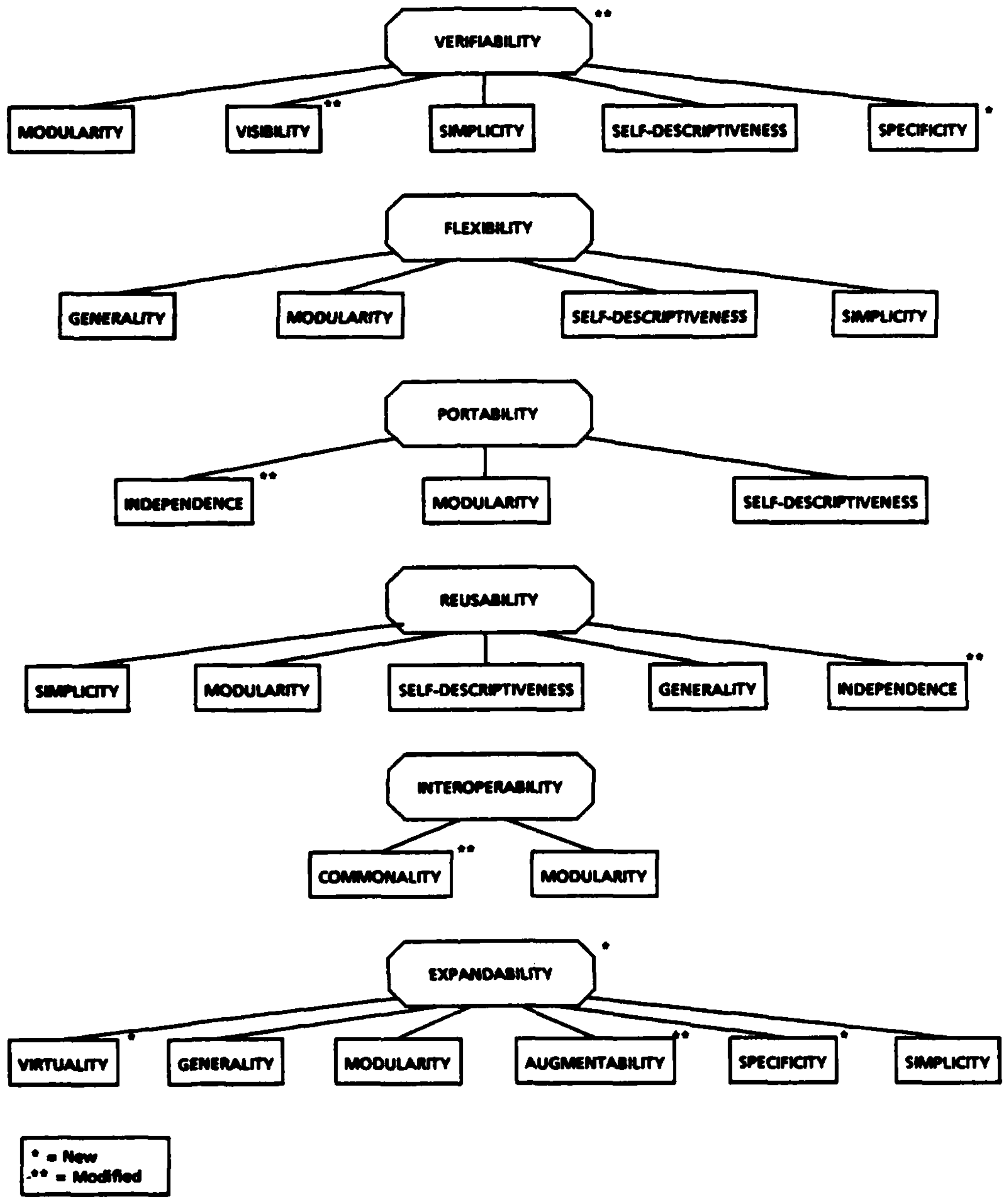


Figure 3.1-1 Relationship of Criteria to Software Quality Factors (Continued)

Table 3.1-8 Software Quality Factors and Criteria

QUALITY CRITERIA (SOFTWARE OR SOFTWARE PRODUCTION OREINTED) \ QUALITY FACTOR (USER ORIENTED)	SOFTWARE OPERATION						SOFTWARE REVISION			SOFTWARE TRANSITION			
	CORRECTNESS	RELIABILITY	EFFICIENCY	INTEGRITY	USABILITY	SURVIVABILITY*	MAINTAINABILITY	VERIFIABILITY**	FLEXIBILITY	PORTABILITY	REUSABILITY	INTEROPERABILITY	EXPANDABILITY*
● ACCURACY		X											
● ANOMALY MANAGEMENT**		X				$\boxed{X}$							
● AUGMENTABILITY**													$\boxed{X}$
● AUTONOMY*						$\boxed{X}$							
● COMMONALITY**												X	
● COMMUNICATIVENESS					X								
● COMPLETENESS	X												
● CONCISENESS							X						
● CONSISTENCY	X	X					X						
● DISTRIBUTEDNESS*						$\boxed{X}$							
● EFFECTIVENESS**			X										
● GENERALITY									X		X		$\boxed{X}$
● INDEPENDENCE**										X	X		
● MODULARITY						$\boxed{X}$	X	X	X	X	X	X	$\boxed{X}$
● OPERABILITY					X								
● RECONFIGURABILITY*						$\boxed{X}$							
● SELF-DESCRIPTIVENESS							X	X	X	X	X		
● SIMPLICITY	$\boxed{X}$	X					X	X	$\boxed{X}$		$\boxed{X}$		$\boxed{X}$
● SPECIFICITY*	$\boxed{X}$							$\boxed{X}$					$\boxed{X}$
● SYSTEM ACCESSIBILITY**				X									
● TRACEABILITY	X												
● TRAINING					X								
● VIRTUALITY*				$\boxed{X}$	$\boxed{X}$								$\boxed{X}$
● VISIBILITY**					$\boxed{X}$		$\boxed{X}$	X					

* = NEW
** = DIFFERENT

$\boxed{X}$ = NEW RELATIONSHIP
X = PREVIOUSLY ESTABLISHED RELATIONSHIP

Table 3.1-9 Software Quality Criteria Definitions

CRITERION	DEFINITION	RELATED FACTORS
• ACCURACY	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE THE REQUIRED PRECISION IN CALCULATIONS AND OUTPUTS.	• RELIABILITY
• ANOMALY MANAGEMENT**	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE FOR CONTINUITY OF OPERATIONS UNDER AND RECOVERY FROM NON-NOMINAL CONDITIONS.	• RELIABILITY • SURVIVABILITY*
• AUGMENTABILTIY*	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE FOR EXPANSION OF CAPABILITY FOR FUNCTIONS AND DATA.	• EXPANDABILITY*
• AUTONOMY*	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH DETERMINE ITS NON-DEPENDENCY ON INTERFACES AND FUNCTIONS.	• SURVIVABILITY*
• COMMONALITY**	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE FOR THE USE OF INTERFACE STANDARDS FOR PROTOCOLS, ROUTINES, AND DATA REPRESENTATIONS.	• INTEROPERABILITY
• COMMUNICATIVENESS	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE USEFUL INPUTS AND OUTPUTS WHICH CAN BE ASSIMULATED.	• USABILITY
• COMPLETENESS	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE FULL IMPLEMENTATION OF THE FUNCTIONS REQUIRED.	• CORRECTNESS
• CONCISENESS	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE FOR IMPLEMENTATION OF A FUNCTION WITH A MINIMUM AMOUNT OF CODE.	• MAINTAINABILITY
• CONSISTENCY	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE FOR UNIFORM DESIGN AND IMPLEMENTATION TECHNIQUES AND NOTATION.	• CORRECTNESS • MAINTAINABILITY • RELIABILITY
• DISTRIBUTEDNESS*	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH DETERMINE THE DEGREE TO WHICH SOFTWARE FUNCTIONS ARE GEOGRAPHICALLY OR LOGICALLY SEPARATED WITHIN THE SYSTEM.	• SURVIVABILITY*
• EFFECTIVENESS**	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE FOR MINIMUM UTILIZATION OF RESOURCES (PROCESSING TIME, STORAGE, OPERATOR TIME) IN PERFORMING FUNCTIONS.	• EFFICIENCY
• GENERAILITY	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE BREADTH TO THE FUNCTIONS PERFORMED WITH RESPECT TO THE APPLICATION.	• FLEXIBILITY • EXPANDABILITY* • REUSABILITY
• INDEPENDENCE**	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH DETERMINE ITS NON-DEPENDENCY ON THE SOFTWARE ENVIRONMENT (COMPUTING SYSTEM, OPERATING SYSTEM, UTILITIES, INPUT/OUTPUT ROUTINES, LIBRARIES).	• PORTABILITY • REUSABILITY

* = NEW
** = MODIFIED

Table 3.1-9 Software Quality Criteria Definitions (Continued)

CRITERION	DEFINITION	RELATED FACTORS
• MODULARITY	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE A STRUCTURE OF HIGHLY COHESIVE MODULES WITH OPTIMUM COUPLING.	•PORTABILITY •REUSABILITY
• OPERABILITY	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH DETERMINE OPERATIONS AND PROCEDURES CONCERNED WITH THE OPERATION OF THE SOFTWARE.	•SURVIVABILITY* •FLEXIBILITY •INTEROPERABILITY •MAINTAINABILITY •PROTABILITY •EXPANDABILITY* •VERIFIABILITY** •REUSABILITY
• RECONFIGURABILITY*	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE FOR CONTINUITY OF SYSTEM OPERATION WHEN ONE OR MORE PROCESSORS, STORAGE UNITS, OR COMMUNICATION LINKS FAILS.	•USABILITY
• SELF-DESCRIPTIVENESS	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE EXPLANATION OF THE IMPLEMENTATION OF A FUNCTION.	•MAINTAINABILITY •FLEXIBILITY •REUSABILITY •VERIFIABILITY** •PORTABILITY
• SIMPLICITY	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE FOR THE DEFINITION AND IMPLEMENTATION OF FUNCTIONS IN THE MOST NON-COMPLEX AND UNDERSTANDABLE MANNER.	•RELIABILITY •FLEXIBILITY •MAINTAINABILITY •REUSABILITY •VERIFIABILITY** •EXPANDABILITY*
• SPECIFICITY*	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE FOR SINGULARITY IN THE DEFINITION AND IMPLEMENTATION OF FUNCTIONS.	•CORRECTNESS •EXPANDABILITY* •VERIFIABILITY**
• SYSTEM ACCES-SIBILITY**	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE FOR CONTROL AND AUDIT OF ACCESS TO THE SOFTWARE AND DATA.	•INTEGRITY
• TRACEABILITY	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE A THREAD OF ORIGIN FROM THE IMPLEMENTATION TO THE REQUIREMENTS WITH RESPECT TO THE SPECIFIED DEVELOPMENT ENVELOPE AND OPERATIONAL ENVIRONMENT.	•CORRECTNESS
• TRAINING	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE TRANSITION FROM CURRENT OPERATION OR PROVIDE INITIAL FAMILIARIZATION.	•USABILITY
• VIRTUALITY*	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PRESENT A SYSTEM THAT DOES NOT REQUIRE USER KNOWLEDGE OF THE PHYSICAL, LOGICAL, OR TOPOLOGICAL CHARACTERISTICS (E.G., NUMBER OF PROCESSORS/DISKS, STORAGE LOCATIONS).	•INTEGRITY •USABILITY •EXPANDABILITY*
• VISIBILITY**	• THOSE ATTRIBUTES OF THE SOFTWARE WHICH PROVIDE STATUS MONITORING OF THE DEVELOPMENT AND OPERATION (E.G., INSTRUMENTATION).	•USABILTY •MAINTAINABILITY •VERIFIABILITY**

* = NEW
** = MODIFIED

3.1.2.1 Revised Criteria

Of the twenty three previously identified criteria, twelve remain identical, ten were revised and consolidated into seven criteria with new names, and one was changed to a quality factor (expandability - formerly a criterion of flexibility). The revised criteria are discussed below.

Error tolerance was changed to the more general term "anomaly management" which includes management of the system configuration under non-nominal conditions - a common task for distributed system software. Execution efficiency and storage efficiency were consolidated to the more general term effectiveness, meaning minimum utilization of resources in performing functions. This enables the inclusion of operator time in the measure of efficiency. Access control and access audit were consolidated into one term--accessibility. Instrumentation was changed to the more inclusive term visibility. This enables consideration of tools and procedures used to monitor the status of the development process in addition to tools for monitoring the status of product operation.

Software system independence and machine independence were replaced by the term independence. This enables consideration of modular and functional independence. Communications commonality and data commonality were combined in the term commonality. Expandability was changed from a criterion of flexibility to a quality factor--providing the capability to consider system growth at the quality factor level.

3.1.2.2 New Criteria

Five new criteria were identified during the investigation of software characteristics for distributed systems as indicated in Table 3.1-8: autonomy, distributedness, reconfigurability, specificity, and virtuality. Although these new criteria were identified through concerns for distributed systems, specificity and virtuality may also be applicable to uniprocessor systems. Reconfigurability, distributedness, and autonomy are associated only with the quality factor survivability and may be applicable to very few uniprocessor systems.

3.1.2.3 Effect of Criteria on Software Quality Factors

The effect of each criterion on each quality factor was evaluated; the results are summarized in Table 3.1-10. The criteria indicating the presence of a quality factor are indicated with an "X". If the existence of the attributes characterized by each criterion positively impacts a factor, an "△" was placed in the matrix. If the existence of the attributes characterized by each criterion negatively impacts a factor, an "▲" was placed in the matrix. If there was no relationship between the criterion and the quality factor or if the relationship was highly application-dependent, the matrix was left blank. These criteria-to-factor relationships are the basis for the factor-to-factor relationships discussed in Section 3.1.1.

3.1.3 Software Quality Metrics

Previous analysis and evaluation had resulted in the identification of thirty nine metrics. Eleven new metrics were identified under this contract. These metrics are:

o	AM.6	Communications Errors Checklist
o	AM.7	Node/Communications Failures Checklist
o	AG.3	Channel Extensibility Measure
o	AG.4	Design Extensibility Checklist
o	AU.1	Interface Complexity Measure
o	AU.2	Self-Sufficiency Checklist
o	DI.1	Design Structure Checklist
o	MO.3	Modular Design Measure
o	RE.1	Restructure Checklist
o	SP.1	Scope of Function Measure
o	VR.1	System/Data Independence Checklist

The complete list of metrics is shown in Table 3.1-11. Some of the metric names were revised as was shown in Table 2.2-2.

Table 3.1-10 Effect of Criteria on Software Quality Factors

QUALITY FACTOR / QUALITY CRITERIA	SOFTWARE OPERATION						SOFTWARE REVISION			SOFTWARE TRANSITION			
	CORRECTNESS	RELIABILITY	EFFICIENCY	INTEGRITY	USABILITY	SURVIVABILITY*	MAINTAINABILITY	VERIFIABILITY*	FLEXIBILITY	PORTABILITY	REUSABILITY	INTEROPERABILITY	EXPANDABILITY*
• ACCURACY		x	▲		△								
• ANOMALY MANAGEMENT**	△	x	▲		△	x							
• AUGMENTABILITY**									△				x
• AUTONOMY*						x							
• COMMONALITY**		△		▲							△	x	
• COMMUNICATIVENESS			▲		x		△	△	△		△		△
• COMPLETENESS	x	△			△	△							
• CONCISENESS	△		△				x	△					
• CONSISTENCY	x	x					x	△	△		△		△
• DISTRIBUTEDNESS*				▲		x							
• EFFECTIVENESS**			x				▲	▲		▲			
• GENERALITY		▲	▲	▲					x		x	△	x
• INDEPENDENCE**			▲			△			△	x	x		△
• MODULARITY			▲			x	x	x	x	x	x	x	x
• OPERABILITY			▲		x								
• RECONFIGURABILITY*		△	▲			x			▲	▲	▲		
• SELF-DESCRIPTIVENESS			▲				x	x	x	x	x		△
• SIMPLICITY	x	x	△				x	x	x		x		x
• SPECIFICITY*	x	△	△				△	x					x
• SYSTEM ACCESSIBILITY**			▲	x	△				▲			▲	
• TRACEABILITY	x						△	△	△		△		△
• TRAINING					x						△		
• VIRTUALITY*			▲	x	x								x
• VISIBILITY**			▲		x		x	x					

* = NEW
** = MODIFIED

△ = POSITIVE CORRELATION
▲ = NEGATIVE CORRELATION

X = BASIC ASSOCIATION

TABLE 3.1-11. METRIC TABLES SUMMARY

CRITERIA	ACRONYM	METRICS
ACCURACY	AY.1	ACCURACY CHECKLIST
ANOMALY MANAGEMENT	AM.1	ERROR TOLERANCE/CONTROL CHECKLIST
	AM.2	IMPROPER INPUT DATA CHECKLIST
	AM.3	COMPUTATIONAL FAILURES CHECKLIST
	AM.4	HARDWARE FAULTS CHECKLIST
	AM.5	DEVICE ERRORS CHECKLIST
	AM.6*	COMMUNICATION ERRORS CHECKLIST
	AM.7*	NODE/COMMUNICATIONS FAILURES CHECKLIST
AUGMENTABILITY	AG.1	DATA STORAGE EXPANSION MEASURE
	AG.2	COMPUTATION EXTENSIBILITY MEASURE
	AG.3*	CHANNEL EXTENSIBILITY MEASURE
	AG.4*	DESIGN EXTENSIBILITY CHECKLIST
AUTONOMY*	AU.1*	INTERFACE COMPLEXIBILITY MEASURE
	AU.2*	SELF-SUFFICIENCY CHECKLIST
COMMONALITY	CL.1	COMMUNICATIONS COMMONALITY CHECK-LIST
	CL.2	DATA COMMONALITY CHECKLIST
COMMUNICATIVENESS	CM.1	USER INPUT INTERFACE MEASURE
	CM.2	USER OUTPUT INTERFACE MEASURE
COMPLETENESS	CP.1	COMPLETENESS CHECKLIST
CONCISENESS	CO.1	HALSTEAD'S MEASURE
CONSISTENCY	CS.1	PROCEDURE CONSISTENCY MEASURE
	CS.2	DATA CONSISTENCY MEASURE
DISTRIBUTEDNESS*	DI.1*	DESIGN STRUCTURE CHECKLIST
EFFECTIVENESS	EF.1	PERFORMANCE REQUIREMENTS
	EF.2	ITERATIVE PROCESSING EFFICIENCY MEASURE
	EF.3	DATA USAGE EFFICIENCY MEASURE
	EF.4	STORAGE EFFICIENCY MEASURE

* = New

TABLE 3.1-11. METRIC TABLES SUMMARY (Continued)

CRITERIA	ACRONYM	METRICS
GENERALITY	GE.1 GE.2	MODULE REFERENCE BY OTHER MODULES IMPLEMENTATION FOR GENERALITY CHECKLIST
INDEPENDENCE	ID.1 ID.2	SOFTWARE SYSTEM INDEPENDENCE MEASURE MACHINE INDEPENDENCE MEASURE
MODULARITY	MO.1 MO.2 MO.3*	INDEPENDENCE/STABILITY MEASURE MODULAR IMPLEMENTATION MEASURE MODULAR DESIGN MEASURE
OPERABILITY	OP.1	OPERABILITY CHECKLIST
RECONFIGURABILITY*	RE.1*	RESTRUCTURE CHECKLIST
SELF-DESCRIPTIVENESS	SD.1 SD.2 SD.3	QUANTITY OF COMMENTS EFFECTIVENESS OF COMMENTS MEASURE DESCRIPTIVENESS OF LANGUAGE MEASURE
SIMPLICITY	SI.1 SI.2 SI.3 SI.4	DESIGN STRUCTURE MEASURE STRUCTURED LANGUAGE OR PRE-PROCESSOR DATA AND CONTROL FLOW COMPLEXITY MEASURE CODING SIMPLICITY MEASURE
SPECIFICITY*	SP.1*	SCOPE OF FUNCTION MEASURE
SYSTEM ACCESSIBILITY	SA.1 SA.2	ACCESS CONTROL CHECKLIST ACCESS AUDIT CHECKLIST
TRACEABILITY	TR.1	CROSS REFERENCE
TRAINING	TN.1	TRAINING CHECKLIST
VIRTUALITY*	VR.1*	SYSTEM/DATA INDEPENDENCE CHECKLIST
VISIBILITY	VS.1 VS.2 VS.3	MODULE TESTING MEASURE INTEGRATION TESTING MEASURE SYSTEM TESTING MEASURE

* = New

3.2 SYSTEM QUALITY FRAMEWORK

Part of the task of the acquisition manager is to analyze system quality requirements and to determine the suballocation of those quality requirements to the software subsystem. To aid in this task for distributed system applications a set of system-level quality factors was defined, the corresponding quality criteria were identified, and a correspondence between distributed system quality and distributed system software quality was developed. The following paragraphs discuss the factors, the criteria, and their correspondence with software quality. There is a surface similarity among many of the system-level factors and criteria and the software factors and criteria. This is understandable since the software is viewed as a complete subsystem, and most of the factors and criteria are general. However, the correspondence developed between system and software quality factors illustrates the distinction between the system view and the software view and emphasizes the uniqueness of each factor.

3.2.1 System Quality Factors

Sixteen system quality factors have been identified. Table 3.2-1 lists these quality factors and poses a question which indicates the relevancy of each factor to a user. Previous analysis and evaluation of software quality factors resulted in grouping factors into three categories: product operation, product revision, and product transition. This scheme was also used to group system quality factors as it emphasizes the user's view of system life-cycle management. Table 3.2-2 provides definitions for all of the system quality factors.

Twelve of the sixteen system quality factors are similar to the software quality factors (see Section 3.1). Four of the quality factors are unique to systems or complete subsystems: availability, safety, transportability, and interchangeability. Availability requirements are usually allocated down to the computational subsystem as a whole including the software. Safety requirements are satisfied through design and construction and throu[illegible] additional system functions, sometimes supported by software. Transportability refers to the physical relocation of a system, and interchangeability refers to the transfer of a system component for use in another configuration. Both transportability and interchangeability are especially applicable to fielded military systems. The software factor portability is similar in nature to these two factors.

Table 3.2-1 System Quality Factor Identification

Activity	User Concern	Quality Factor
PRODUCT OPERATION	DOES IT DO WHAT IT'S SUPPOSED TO?	CORRECTNESS
	WHAT CONFIDENCE CAN BE PLACED IN WHAT IT DOES?	RELIABILITY
	HOW WELL DOES IT UTILIZE THE RESOURCES?	EFFICIENCY
	HOW SECURE IS IT?	INTEGRITY
	HOW EASY IS IT TO USE?	USABILITY
	HOW MUCH OF THE TIME CAN IT BE USED?	AVAILABILITY
	HOW SAFE IS IT?	SAFETY
	HOW WELL WILL IT PERFORM UNDER ADVERSE CONDITIONS?	SURVIVABILITY
	HOW EASILY CAN IT BE RELOCATED?	TRANSPORTABILITY
PRODUCT REVISION	CAN IT BE REPAIRED?	MAINTAINABILITY
	CAN ITS OPERATION AND PERFORMANCE BE VERIFIED?	VERIFIABILITY
	CAN IT BE CHANGED?	FLEXIBILITY
PRODUCT TRANSITION	CAN IT BE USED IN ANOTHER ENVIRONMENT?	INTERCHANGE-ABILITY
	CAN IT BE USED IN ANOTHER APPLICATION?	REUSABILITY
	CAN IT BE INTERFACED WITH ANOTHER SYSTEM?	INTEROPERABILITY
	CAN ITS CAPABILITY OR PERFORMANCE BE EXPANDED OR UPGRADED?	EXPANDABILITY

Table 3.2-2 System Quality Factor Definitions

Activity	Quality Factor	Definition
OPERATION	CORRECTNESS	EXTENT TO WHICH THE SYSTEM SATISFIES ITS SPECIFICATIONS AND FULFILLS THE USER'S MISSION OBJECTIVES.
	RELIABILITY	PROBABILITY THAT THE SYSTEM WILL PERFORM ITS LOGICAL OPERATIONS IN THE SPECIFIED ENVIRONMENT WITHOUT FAILURE.
	EFFICIENCY	DEGREE OF UTILIZATION OF RESOURCES IN PERFORMING FUNCTIONS.
	INTEGRITY	EXTENT TO WHICH UNAUTHORIZED ACCESS TO THE SYSTEM OR SYTEM INFORMATION CAN BE CONTROLLED.
	USABILITY	EFFORT FOR TRAINING AND SYSTEM OPERATION.
	AVAILABILITY	PORTION OF THE TOTAL OPERATIONAL TIME THAT THE SYSTEM PERFORMS OR SUPPORTS CRITICAL FUNCTIONS.
	SAFETY	PROBABILITY THAT THE SYSTEM WILL NOT CAUSE DAMAGE OR PHYSICAL INJURY.
	SURVIVABILITY	PROBABILITY THAT THE SYSTEM WILL CONTINUE TO PERFORM OR SUPPORT CRITICAL FUNCTIONS WHEN A PORTION OF THE SYSTEM IS INOPERABLE.
	TRANSPORTABILITY	EFFORT TO PHYSICALLY RELOCATE THE SYSTEM.
REVISION	MAINTAINABILITY	AVERAGE EFFORT TO LOCATE AND FIX A SYSTEM FAILURE.
	VERIFIABILITY	EFFORT TO VERIFY THE SPECIFIED SYSTEM OPERATION AND PERFORMANCE.
	FLEXIBILITY	EFFORT TO EXTEND THE SYSTEM MISSIONS TO SATISFY OTHER REQUIREMENTS.
TRANSITION	INTERCHANGEABILITY	EFFORT TO TRANSFER A SYSTEM COMPONENT FOR USE IN ANOTHER OPERATING ENVIRONMENT (E.G., CONFIGURATION).
	REUSABILITY	EFFORT TO CONVERT A SYSTEM COMPONENT FOR USE IN ANOTHER APPLICATION
	INTEROPERABILITY	EFFORT REQUIRED TO COUPLE THE SYSTEM WITH ANOTHER SYSTEM.
	EXPANDABILITY	EFFORT TO INCREASE SYSTEM CAPABILITY OR PERFORMANCE - ENHANCE CURRENT FUNCTIONS OR ADD NEW FUNCTIONS.

3.2.2 System Quality Criteria

Thirty two quality criteria have been identified for distributed systems as illustrated in Figure 3.2-1 and Table 3.2-3. Twenty-four of the criteria are the same criteria as those identified for distributed system software. Eight are peculiar to distributed systems: self-containedness, homogeneity, compliance, validity, clarity, comprehensibility, supportability, and compatibility.

3.2.3 System Quality and Software Quality Correspondence

The following paragraphs discuss the software quality factors which are required if a high quality rating has been specified for one of the system quality factors. These relationships aid the acquisition manager in the allocation of system quality requirements to the software subsystem.

The discussions may also aid the acquisition manager in prioritizing or weighting the software quality criteria. For example, the discussion of system survivability points out the importance of anomaly management in ensuring system survivability; this in turn places a large emphasis on the anomaly management criterion of the software quality factor survivability. Another example is in the discussion of system reusability. A high quality rating for system reusability implies software reusability, flexibility, and verifiability. These three software quality factors have three criteria in common: modularity, self-descriptiveness, and simplicity. This would place added emphasis on these criteria over the other criteria of these three quality factors.

Table 3.2-4 summarizes the correspondence between system and software quality factors. A positive relationship is indicated with an (x). A relationship that is application dependent is indicated with a (-).

High system correctness implies both high software correctness and high software verifiability. The ability to verify software operation and performance against the specifications and mission objectives aids in ensuring the correctness of the overall system.

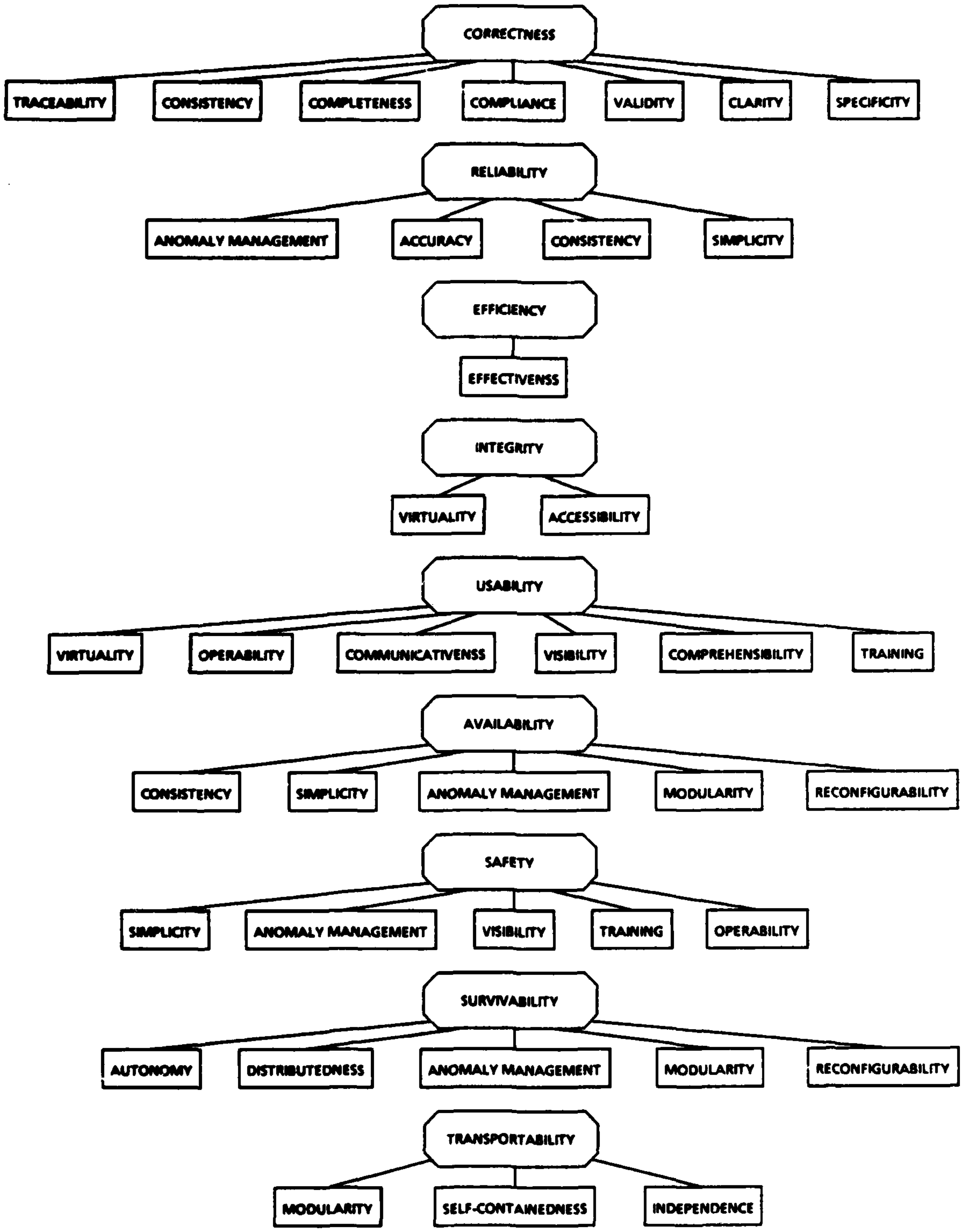


Figure 3.2-1 Relationship of Criteria to System Quality Factors

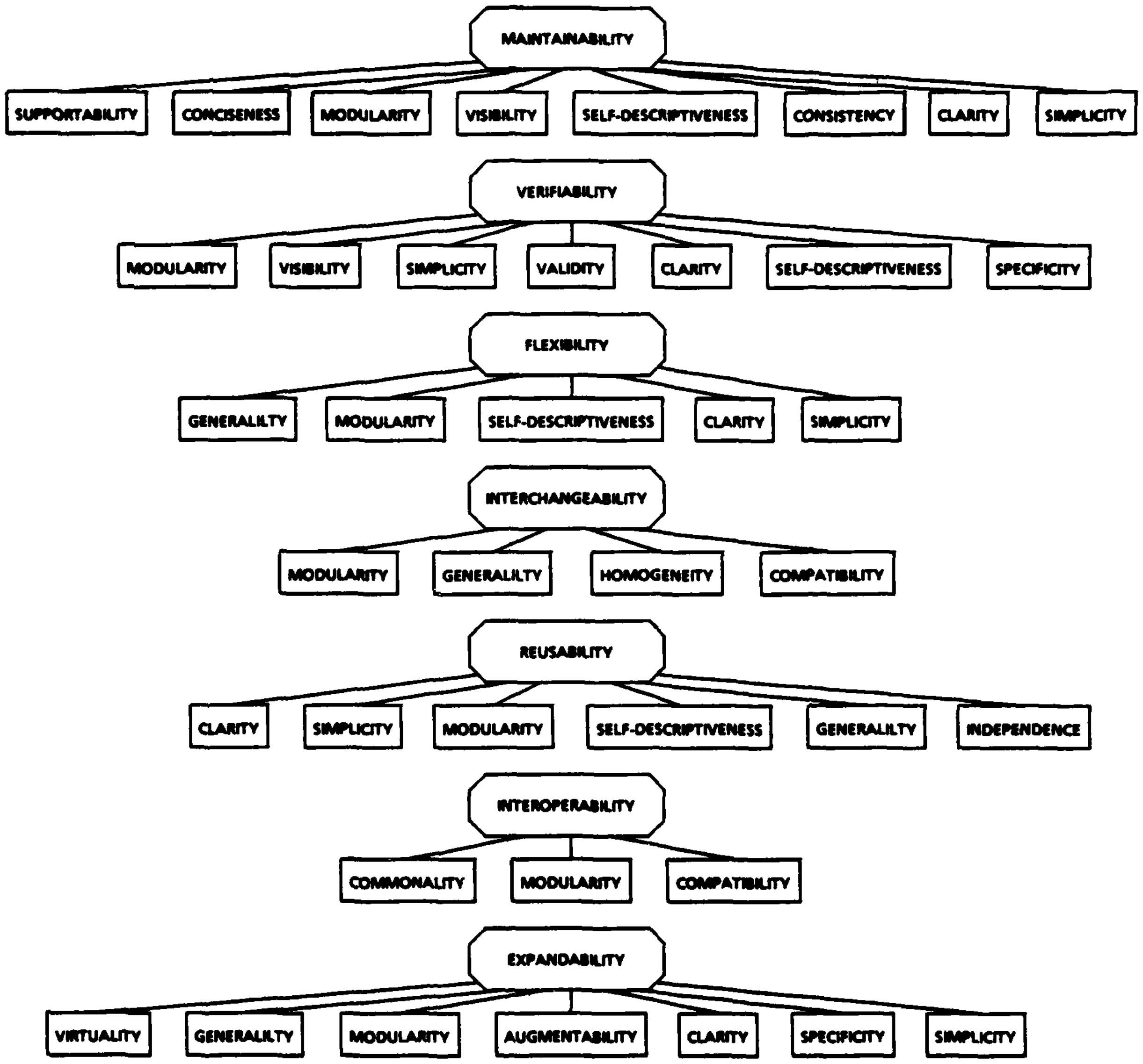


Figure 3.2-1 Relationship of Criteria to System Quality Factors (Continued)

Table 3.2-3 Distributed System Quality Factors and Criteria

QUALITY FACTOR (USER ORIENTED) / QUALITY CRITERIA (SYSTEM OR SYSTEM PRODUCTION OREINTED)	SYSTEM OPERATION									SYSTEM REVISION			SYSTEM TRANSITION			
	CORRECTNESS	RELIABILITY	EFFICIENCY	INTEGRITY	USABILITY	AVAILABILITY	SAFETY	SURVIVABILITY	TRANSPORTABILITY	MAINTAINABILITY	VERIFIABILITY	FLEXIBILITY	INTERCHANGEABILITY	REUSABILITY	INTEROPERABILITY	EXPANDABILITY
• TRACEABILITY	X															
• CONSISTENCY	X	X				X				X						
• COMPLETENESS	X															
• COMPLIANCE	X															
• VALIDITY	X										X					
• CLARITY	X									X	X	X		X		X
• SPECIFICITY	X										X					X
• SIMPLICITY	X	X				X	X			X	X	X		X		X
• ANOMALY MANAGEMENT		X				X	X	X								
• ACCURACY		X														
• EFFECTIVENESS			X													
• ACCESSIBILITY				X												
• VIRTUALITY				X	X											X
• VISIBIILTY					X		X			X	X					
• COMPREHENSIBILITY					X											
• TRAINING					X		X									
• COMMUNICATIVENESS					X											
• OPERABILITY					X		X									
• MODULARITY						X		X	X	X	X	X	X	X	X	X
• RECONFIGURABILITY						X		X								
• DISTRIBUTEDNESS								X								
• AUTONOMY								X								
• SELF-CONTAINEDNESS									X							
• CONCISENESS										X						
• SUPPORTABILITY										X						
• SELF-DESCRIPTIVENESS										X	X	X		X		
• GENERALITY												X	X	X		X
• HOMOGENEITY													X			
• INDEPENDENCE									X					X		
• AUGMENTABILITY																X
• COMPATIBILITY													X		X	
• COMMONALITY															X	

Table 3.2-4 Relationship between System and Software Factors

SYSTEM QUALITY FACTORS	SOFTWARE QUALITY FACTORS	OPERATION						REVISION			TRANSITION			
		CORRECTNESS	RELIABILITY	EFFICIENCY	INTEGRITY	USABILITY	SURVIVABILITY	MAINTAINABILITY	VERIFIABILITY	FLEXIBILITY	PORTABILITY	REUSABILITY	INTEROPERABILITY	EXPANDABILITY
OPERATION	● CORRECTNESS	X							X					
	● RELIABILITY	X	X		X									
	● EFFICIENCY			X		X								
	● INTEGRITY				X		–							
	● USABILITY		–			X								
	● AVAILABILITY		X				X	X						
	● SAFETY	X	X		X				X					
	● SURVIVABILITY				X		X						X	
	● TRANSPORTABILITY		X	X	X	X								
REVISION	● MAINTAINABILITY						X	X						
	● VERIFIABILITY						X	X	X					
	● FLEXIBILITY				–				X	X				
TRANSITION	● INTERCHANGEABILITY									X	X	X		
	● REUSABILITY								X	X		X		
	● INTEROPERABILITY										–	X	X	
	● EXPANDABILITY			–					X	X				X

X = POSITIVE RELATIONSHIP

– = APPLICATION DEPENDENT

High system reliability implies high software reliability, correctness, and integrity. Both software reliability and software correctness contribute to the ability of the system to perform intended functions. High software integrity insures that system reliability will not be adversely effected by accidental or deliberate unauthorized access to the software or the data.

High system efficiency implies both high software efficiency and high software usability. Software usability directly affects operator effectiveness and efficiency; and the system operator is a factor in the measure of system efficiency.

High system integrity implies high software integrity. In most applications system integrity is dependent upon the software and continued software functioning. In these applications software survivability would also impact system integrity.

High system usability implies high software usability. In applications where accuracy and precision affect the amount of time and effort to operate the system, the quality criteria accuracy would also be emphasized. (Accuracy is an attribute of the quality factor reliability.)

High system availability implies high software reliability, survivability, and maintainability. High quality ratings for these factors ensure that the system will seldom fail, that critical functions will continue to be performed in the event of a failure, and that the fault will be quickly corrected.

High system safety implies high software correctness, reliability, integrity, and verifiability. High ratings for these factors ensure that the system will perform as specified, that it will seldom fail, and that it is secure from unauthorized access.

High system survivability implies redundancy of components and communication paths and implies complex anomaly management. Complex anomaly management places emphasis on high software survivability. Emphasis is also placed on high software interoperability for redundancy and increased communications and, for networks with a variety of users, on high software integrity because of the increased vulnerability to unauthorized access.

High system <u>transportability</u> implies low power, light weight, and compactness. These result in constraints on the computing system such as: limited storage; emphasis on firmware rather than software; limited facilities for data entry and display; and wireless communication. These constraints place emphasis on the software quality factors of efficiency, integrity, and usability. Maintenance costs for the software of a transportable system would naturally be high. This in turn places emphasis on the software quality factor reliability to reduce the probability of failure.

High system <u>maintainability</u> is enhanced by software which is easily maintainable and by software which aids in the detection and isolation of faults. This places emphasis on high software maintainability and on high software survivability to continue fault detection and isolation functions even when a portion of the system is inoperable.

High system <u>verifiability</u> implies component modularity, function modularity, fault isolation, high visibility of system operation through instrumentation and system displays, and diagnostic aids such as self-test capabilities. This places emphasis on high software verifiability and high software maintainability. Also, high software survivability would enable functions such as instrumentation, displays, and self-test to continue when a portion of the system is inoperable.

High system <u>flexibility</u> implies modular system components and generality of component functions. This requires flexible software which is both modular and general and emphasizes verification of changes. In addition to high flexibility and verifiability, high integrity would also be emphasized in instances where modification of functions, missions, or data could possibly compromise security.

High system <u>interchangeability</u> implies that a family of systems (or subsystems) has components which are similar in function and which may be substituted for each other. This implies that there may be a need to reuse software from system to system (high reusability), to transfer software to another system configuration (high portability), or to modify software missions, functions, or data (high flexibility).

High system <u>reusability</u> implies modularity of components and modularity and specificity of functions. This eases the task of selection and removal for reuse. It also implies that functions and components are general enough to be tailored to a new application. This places emphasis on high software reusability, on high software flexibility to accommodate changes, and on high software verifiability to test changes.

High system interoperability implies commonality of interface protocols, routines, and data representations. It also implies compatibility of interface equipment. This places emphasis on the high interoperability of the software and the ability to reuse the software on interfacing systems (high reusability). For some applications there may also be a need to transfer software to an interfacing system (portability).

High system expandability implies generality and modularity of components and functions and implies spare system capacity. This emphasizes high software expandability, high software flexibility to incorporate enhancements and new functions, and high software verifiability to test changes. For a system which is capacity limited, high software efficiency would be emphasized.

4.0
TRANSITIONAL STUDY

This section describes the software quality framework elements which were proposed but dropped from the framework. One quality factor was dropped. Six quality criteria were dropped. No metrics were dropped; but 26 metric elements were dropped.

4.1 FACTORS STUDIED AND DROPPED

The quality factor evolvability was proposed and dropped. Evolvability is defined as the extent to which the software performance can be enhanced by the incorporation of new technology (e.g., algorithm, compiler). The criteria associated with the quality factor are: virtuality, generality, modularity, specificity, and simplicity.

Evolvability was dropped because of its similarity to expandability, and the definition of expandability was modified to include evolvability.

4.2 CRITERIA STUDIED AND DROPPED

Six quality criteria were studied and dropped: clarity, compatibility, compliance, comprehensibility, supportability, and validity. The definitions of these criteria are as follows.

. Clarity	- those attributes of the software which provide non-ambiguous descriptions of functions and implementations.
. Compatibility	- those attributes of the software which provide interface protocols and routines that are appropriate to the interface equipment and features.
. Compliance	- those attributes of the software which promote implementations that conform to the requirements.
. Comprehensibility	- those attributes of the software which enhance understanding of the operation of the software.

. Supportability	- those attributes of the software which provide for ease in creation of new software versions (e.g., use of HOL, version update scheme).
. Validity	- those attributes of the software which constrain implementations to a range of acceptable solutions.

These six criteria were dropped either because of their similarity to and overlap with other criteria or because of lack of metrics and metric elements to support them.

4.3 METRICS STUDIED AND DROPPED

No metrics were dropped from the proposed framework, but twenty six metric elements were studied and dropped. The following metric elements were dropped because of their similarity or overlap with existing metric elements, because they were more applicable to the system than to the software, or because they were insufficiently defined. The metric elements are categorized by applicable criteria.

AUGMENTABILITY

- Spare memory fraction
  1-(bytes of memory used/total bytes available)
- Spare timing (speed capacity)
- Is operating system designed so that functions can easily be added or expanded?
- Are the software functions/modules designed so that the span of control (effect) can be increased?

AUTONOMY

- Atomicity: is each software transaction viewed by users and system as indivisible?
- Built-in-testing.

SIMPLICITY

- Is there, in the code's comments or preamble, a self-contained functional flow diagram (besides text)?
- Are upper and lower bounds for acceptable inputs specified?
- Is the code apparently free of "tricky" code (code which uses little-known side-effects of language features)?
- Is the code structured, in the sense of clear flow of control and lack of go-to's?

DISTRIBUTEDNESS

- Total number kilometers of communication links.
- Diameter: maximum over all nodes of minimum distance between each pair of nodes.
- Number of nodes (processors).
- Number of links.
- Parallelisability (Kuck's metric)
  (Number of independent processors/10 $\log_{10}$ number of independent processors)
- Average redundancy of processors
  (total number of processors/number of independent processors addressed).

RECONFIGURABILITY

- Nonstopability: do nodes run on nonstop processors (multiprocessor redundancy with software control program for graceful degradation)?
- Dynamic backup: can a data base be dumped while it is being used?

- What fraction of functions can be relocated without alteration to other processors?

- Does the system automatically detect if a node has gone off-line?

- File availability: what is the probability that at least one copy of a file resides at a site which is accessible to all other sites?

<u>SPECIFICITY</u>

- Determinism: are the same outputs always produced by the same inputs?

<u>TRAINING</u>

- Is a user disturbed, so as to be less effective, by:
  1) instability, 2) unpredictability, 3) lengthiness of delay, 4) cryptic and artificially terse messages?

- Does the system have a "help" function to answer questions for confused users?

<u>VIRTUALITY</u>

- Virtual memory: does the system use virtual addressing?

- Relational database: is the database management system relational in the sense of E.F. Codd (Communications of the ACM, February 1982, p. 109-117)

5.0
VALIDATION

5.1 VALIDATION OF QUALITY MODEL

5.1.1 Overview of Validation Effort

The two new quality factors, survivability and expandability, were selected for validation. This selection enabled validation coverage of the majority of new framework elements: 100% of new factors, 100% of new criteria, 100% of new metrics, 78% of new metric elements, 80% of new worksheet references, and 78% of new worksheet questions.

The basic steps performed in the validation effort were selecting appropriate projects and modules, collecting metric data, collecting quality rating estimates, and analyzing and correlating metric scores and quality rating values.

The validation approach of RADC-TR-77-369 was followed, wherever possible. But some of the same problems were found as were reported in RADC-TR-77-369 in which various metrics could not be evaluated due to uniformity of development standards which did not permit assessment of the correlation of these metrics with project quality data. (See AM.4 and AM.5 in Table 5.3-1.)

The greatest difficulty in the validation effort was with the quality ratings against which the metrics are to be validated. Quality ratings could only be obtained for factors and criteria at the project or system level, rather than for individual modules. Therefore, the validation effort was necessarily restricted to the internal analysis of metric scores and to the correlations of criteria scores (for criteria within survivability and expandability) with criteria quality ratings from each project. Nevertheless, such correlations show a positive relationship between quality ratings and metric scores, and thus support the addition of the expandability and survivability quality factors to the software quality framework. This result, and the fact that metrics could be successfully collected, also supports the use of the quality metrics methodology with distributed computing systems.

5.1.2 Selection of Projects for Validation

Four Boeing projects were selected for validation; E-3A, UDACS, B-1 Avionics, and the Morgantown Personal Rapid Transit (MPRT) system. Although interviews and preliminary data collection had been conducted earlier on a number of other candidate projects, only these four projects were used for validation.

The four projects, described below, were selected on the basis of the following criteria:

- Distributed computing is integral to design and operation
- Documentation for the entire life-cycle is available
- Project management and software engineering personnel are available for interview
- Survivability and Expandability are significant considerations in system design

For reasons of confidentiality, these four projects, in a permuted order, are identified throughout this final report only by the encrypted designations "A", "B", "C", and "D".

E-3A

E-3A, the AWACS, Airborne Warning and Control System, is a high-capacity radar station and command control station in a Boeing 707 airplane. E-3A software (the AOCP, Airborne Operational Computer Program) resides in two embedded IBM 4PiCC2 computers. The dual 2.1 Mops/sec processors, hot spare arithmetic control units, 3 drums (400 Kwords each), 11 banks of 64K core, and 9 operator consoles are coupled by serial channels through a hard-wired Interface Adapter Unit (which itself has 2 complete backups). Each processor has its own event-driven operating system. E-3A was developed for the USAF, and was modified to an interoperational NATO version. Software in 95% Jovial J-73 and 5% assembly language.

UDACS

UDACS, the Universal Display and Control System, is an airborne radar station and command control station in a U.S. Navy P-3 Orion airplane. Three AYK-14 computers (off-the-shelf CDC 480's also used in the F14) and 4 operator stations are coupled through a dual serial, polled, MIL-STD-1553A bus. Each operator station contains a CRT, 64 K-words memory, and Intel 8086 microprocessor, 96 software programmable switches, keyboard, and trackball. The highly reconfigurable programmable switches each display 12 characters of 12 x 20 pixels as a dense array of LED's built into buttons. 5000 lines of PL/1 make up the HOL code, and a ground support package targets CMS-2 to the 8086's. The Royal New Zealand Airforce is the customer.

B-1 Avionics

The B-1 Avionics operational software resides in two offensive computers and one defensive computer. The computers are linked to sensors, actuator, other computers, and operator consoles through redundant serial buses (1 megabit). The two offensive computers are tightly coupled through a high speed processor link. Some redundant processing is performed, and a degraded mode of operation is possible in the event of failure of one of the offensive computers. The offensive and defensive computers communicate via the redundant serial buses. The source code is written in Jovial J3B and assembly language.

Morgantown (MPRT Phase-II)

The Morgantown operational software resides in two central computers and twelve station computers. There are two computers at each location. The computers are configured in two redundant strings which perform parallel, concurrent processing. Within each string, one central computer communicates with one computer at each of the six stations through 2400 baud modems. The two computers at each location communicate with each other to monitor for failures and to synchronize processing. The source code is written in ANSI STD FORTRAN and assembly language.

5.1.3 Selection of Modules for Validation

E-3A

Modules from E-3A were chosen as follows. One particular CPC was selected which served a communications encoding function. This CPC was in a survivability-critical application and was expected to go through several phases of expansion. Timing information was available for its performance. The modules within this CPC were highly and intricately coupled to each other. All 12 modules in this CPC were selected for data collection and analysis. Although they do not necessarily constitute a typical cross-section of the entire system, they are typical of modules for which distributedness, survivability, and expandability are innate in the design.

UDACS

Modules from UDACS were chosen as follows. One particular CPC was selected which performed a graphics function for data from nearly all the processors in the distributed system. Survivability was reflected in this function as being able to be performed on either of several processors. Expandability was critical, as new functionalities were required to be shown in new display types. The modules in this CPC were highly interconnected. The first 24 modules in this CPC, according to a complete but arbitrary listing, were chosen for data collection and analysis.

B-1 Avionics

The B-1 Avionics operational software consists of two programs — the Offensive Flight Software (OFS) and the Defensive Flight Software (DFS). Modules were chosen from the OFS for validation because this software is distributed between two processors; the DFS resides in one processor. The OFS performs five major functions: navigation, weapon delivery, controls and displays, test, and executive. The modules of the navigation function were chosen for validation because a high percentage of JOVIAL was used and because the functions performed are typical avionics functions.

Morgantown

The Morgantown operational software consists of four separate programs — Central Application Program (CAP), Passenger Station Application Program (PSAP), Maintenance Station Application Program (MSAP), and Executive (EXEC). The modules of CAP were chosen to be used for validation because CAP performs a greater number of functions and more general functions than either PSAP or MSAP and because more FORTRAN was used in the CAP modules, by far, than in the modules of any other program. The major functions performed by CAP modules are: command processing; data collection, recording, and display; communications processing; operations management; and operations monitoring.

5.2 DATA COLLECTION

5.2.1 Metric Worksheets

Specific metric worksheets were developed to collect data for the factors survivability and expandability. These worksheets, shown in Appendix C, list questions whose answers enable computation of values for metric elements. The metric elements are then used to compute metrics for all criteria within the software quality factors of expandability and survivability. The questions are presented in logically related worksheet categories, in order to conceptually simplify the data collection task. There are four consecutively numbered worksheets, one for each of the life-cycle phases in which data is collected:

. Metric Worksheet 1	Requirements Analysis/System Level
. Metric Worksheet 2	Design/System Level
. Metric Worksheet 3	Design/Module Level
. Metric Worksheet 4	Source Code/Module Level

Since there were four Boeing projects selected for analysis a total of 16 worksheets were completed. The number of lines of code analyzed from each project (worksheet 4) were as follows:

Project	Lines of Code
A	3407
B	8589
C	3166
D	2353
Total	17515

5.2.2 Quality Rating Questionnaires

A two-page questionnaire, shown in Appendix D, was developed to gather subjective quality ratings for criteria and factors at the system level. For survivability, respondents were asked to rate the software for the system on a scale of 1 to 10 (ten highest) according to 6 defined "attributes". Five of the attributes are the criteria within survivability, the sixth being survivability itself, although this is not stated on the questionnaire form. This allows for two different values of the survivability rating to be computed. First, the average of the 5 individual criteria ratings may be used. Second, the survivability factor rating itself may be used.

For expandability, the rating questionnaire defines 7 attributes and solicits ratings on a 1 to 10 scale. Six of the attributes are the criteria within expandability, and the seventh is expandability itself. In addition, the questionnaire asks for 2 estimates in terms of manmonths of effort: effort to expand the software for this application and equivalent effort to do the same task from scratch for this application. An expandability quality rating may then be calculated with the formula:

$$R_E = 1 - \frac{\text{Effort to Expand}}{\text{Equivalent Development Effort}}$$

This allows three different values of the expandability rating to be computed. First, the average of the six individual criteria ratings. Second, the expandability factor rating itself. Third, the ratio of effort as defined by the above formula.

Although the values given by questionnaire respondents inevitably contain bias and subjectivity, these negative characteristics were minimized by means of a quasi-delphi technique. That is, several copies of the questionnaire were given out to software managers and software engineers within each project. The average value of the ratings, averaged over all personnel questioned on a given project, can be expected to be more accurate than the values of a single typical respondent. Ideally, the questionnaires should be filled out by personnel who have experience with all four of the surveyed projects, but no such personnel were identified in this study.

A total of 10 questionnaires were completed; three each from projects A, B and D and one from project C.

5.3 DATA ENTRY

5.3.1 Metric Data Entry Files

The handwritten entries from the 16 metric worksheets described in section 5.2.1 (4 projects x 4 life-cycle phases) were entered as raw data files. The raw data files were entered by means of the VI screen editor, and stored on Boeing Aerospace Company's Software Support Center VAX-11/780 computer running under the UNIX operating system.

Raw data files were converted to matrices of worksheet entries by means of small formatting Fortran-77 programs and all data entries were checked against the original worksheets. Data entries were used to calculate metric element and metric values by means of Fortran programs which implemented the metric formulas given in the Metric Tables in Appendix B of Volume II, "Guidebook for Software Quality Measurement". Metric values were also checked for reasonableness; e.g., in one case a computed metric value greater than 1.0 was traced to an incorrect metric definition formula, which was then corrected.

Table 5.3-1 shows a summary of the metric values for all metrics within expandability and survivability, for all four projects, and all four worksheets. The table contains the minima, maxima, standard deviation and mean values for each metric.

Table 5.3-1 Metric Values: All Projects, All Worksheets

METRIC	MIN	MAX	STD.DEV.	MEAN
AG.1	.00	1.00	.38	.57
AG.2	.00	1.00	.38	.59
AG.3	.63	1.00	.17	.89
AG.4	.00	.33	.13	.20
AM.1	.00	.90	.30	.15
AM.2	.00	1.00	.38	.45
AM.3	.00	1.00	.39	.24
AM.4	1.00	1.00	.00	1.00
AM.5	1.00	1.00	.00	1.00
AM.6	.00	1.00	.41	.78
AM.7	.00	1.00	.40	.79
AU.1	.36	1.00	.23	.75
AU.2	.33	.67	.18	.54
DI.1	.25	1.00	.24	.84
GE.1	.07	.50	.24	.22
GE.2	.21	1.00	.28	.74
MO.2	.00	1.00	.30	.28
MO.3	.00	.85	.41	.49
RE.1	.33	1.00	.23	.78
SI.1	.00	1.00	.35	.61
SI.2	.00	1.00	.50	.75
SI.3	.07	.50	.16	.17
SI.4	.56	1.00	.17	.87
SP.1	.40	1.00	.25	.75
VR.1	.00	.67	.46	.56

5.3.2 Quality Rating Entry

The handwritten entries for the 10 Expandability and Survivability Rating Questionnaires, described in section 5.2.2, were entered into the UNIX system in the same manner as the metric worksheet entries described in section 5.3.1. Because of the smaller quantity of data, no formatting programs were needed.

5.3.3 Data Entry in "S"

"S" is a language and system for data analysis which was developed by Richard A. Becker and John M. Chambers of Bell Laboratories, and released in January 1981 to run under the UNIX operating system. Data in S is organized into named datasets which are kept in databases and retrieved automatically when named in an expression. The fundamental data structure is a vector; other data structures such as matrices are built up from vectors. Matrices of metric values and quality ratings were read from their UNIX files into S datasets.

All data analysis described in section 5.4 was accomplished by means of functions in the S language. Parameters for high-level plotting functions on the Hewlett-Packard 7221 pen-plotting terminal were entered, so that the regression graphs shown in Sections 5.4 could be plotted.

5.4 DATA ANALYSIS

5.4.1 Expandability Data Analysis

The data collected from the metric worksheets for expandability related metrics, as described in section 5.2.1, are summarized in Table 5.4-1. This table contains the average score of all metric scores for each criterion of expandability, for each of the four projects. The mean score for expandability is computed as the average of the criteria scores for each project.

Table 5.4-1 EXPANDABILITY CRITERION SCORES

CRITERIA		PROJECT			
		A	B	C	D
AUGMENTABILITY	AG	.51	.52	.60	.46
GENERALITY	GE	.46	.46	.94	.43
MODULARITY	MO	.44	.36	.16	.28
SIMPLICITY	SI	.69	.40	.66	.70
SPECIFICITY	SP	.82	.40	1.00	.77
VIRTUALITY	VR	.67	.67	.50	.50
MEAN CRITERIA SCORE		.60	.47	.64	.52

The data collected from the rating questionnaires described in section 5.2.2 are summarized in Table 5.4-2. This table contains the quasi-delphi quality rating value for each criterion of expandability for each of the four projects. The expandability quality rating is then computed as the average of the criteria ratings (Mean Criteria Rating) for each project. The expandability Factor Rating for each project is the quasi-delphi quality rating value for the attribute expandability, taken from the rating questionnaires. The third expandability rating is calculated in terms of manmonths of effort data taken from the rating questionnaires; this Effort Rating is calculated from the formula

$$R_E = 1 - \frac{\text{Effort to Expand}}{\text{Equivalent Development Effort}}$$

Table 5.4-2 EXPANDABILITY QUALITY RATING VALUES

CRITERIA		PROJECT A	B	C	D
AUGMENTABILITY	AG	.34	.57	.70	.73
GENERALITY	GE	.50	.50	.80	.43
MODULARITY	MO	.47	.80	.65	.63
SIMPLICITY	SI	.53	.67	.35	.27
SPECIFICITY	SP	.80	.70	.65	.50
VIRTUALITY	VR	.30	.53	.75	.77
MEAN CRITERIA RATING		.49	.63	.65	.55
FACTOR RATING		.4	.67	.80	.67
EFFORT RATING (R_E)		.70	.43	.87	.63

The data from Tables 5.4-1 and 5.4-2 was statistically analyzed to compute regression equations and correlation coefficients to compare expandability scores (metrics) with expandability rating values (quality levels). The results of this analysis are shown in Table 5.4-3.

Table 5.4-3 Summary of Statistical Analysis for Expandability

STATISTIC	QUALITY RATING			
	Individual Criteria	Mean Criteria	Factor	Effort (R_E)
Intercept	.61	.62	.66	-.61
Slope	-.05	-.07	.04	2.28
Correlation Coefficient	-.06	-.08	-.02	.96

$Y = a + bX$

a = Intercept	Y = Quality Rating
b = Slope	X = Criteria Score (Metrics)

The data in Tables 5.4-1 and 5.4-2 and the regression/correlation results in Table 5.4-3 are plotted in Figures 5.4-1 through 5.4-4. The X axis represents the expandability criteria scores (calculated from the metric scores) and the Y axis represents the various expandability quality ratings.

The results of this data analysis show poor correlation between expandability criteria scores (metrics) and all of the expandability quality ratings based on individual people rating the system attributes. The results however, show good correlation (0.96) between expandability criteria scores and the expandability quality rating based on the Effort Rating, R_E.

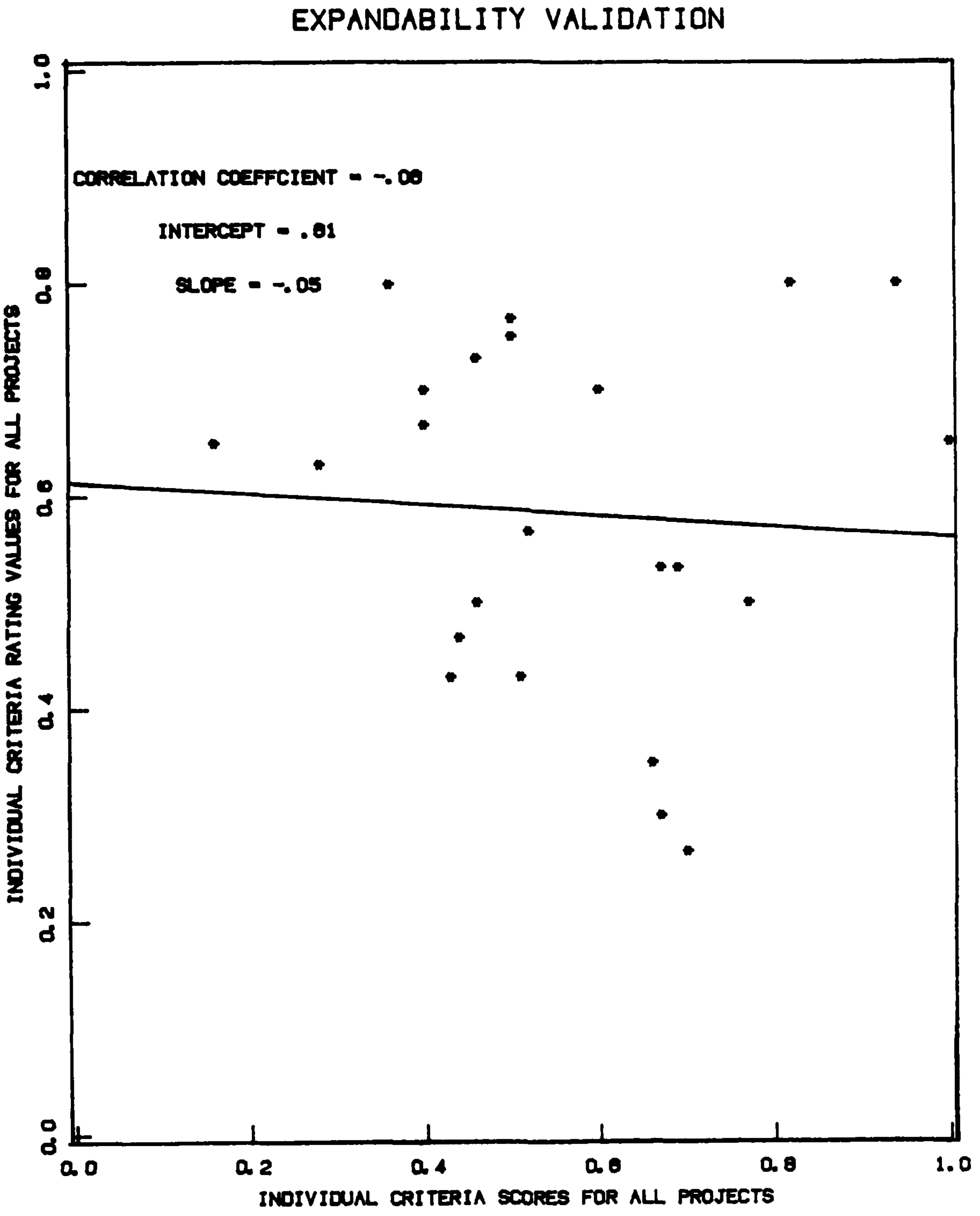


Figure 5.4-1 Expandability Individual Criteria Scores for All Projects vs. Individual Rating Values for All Projects

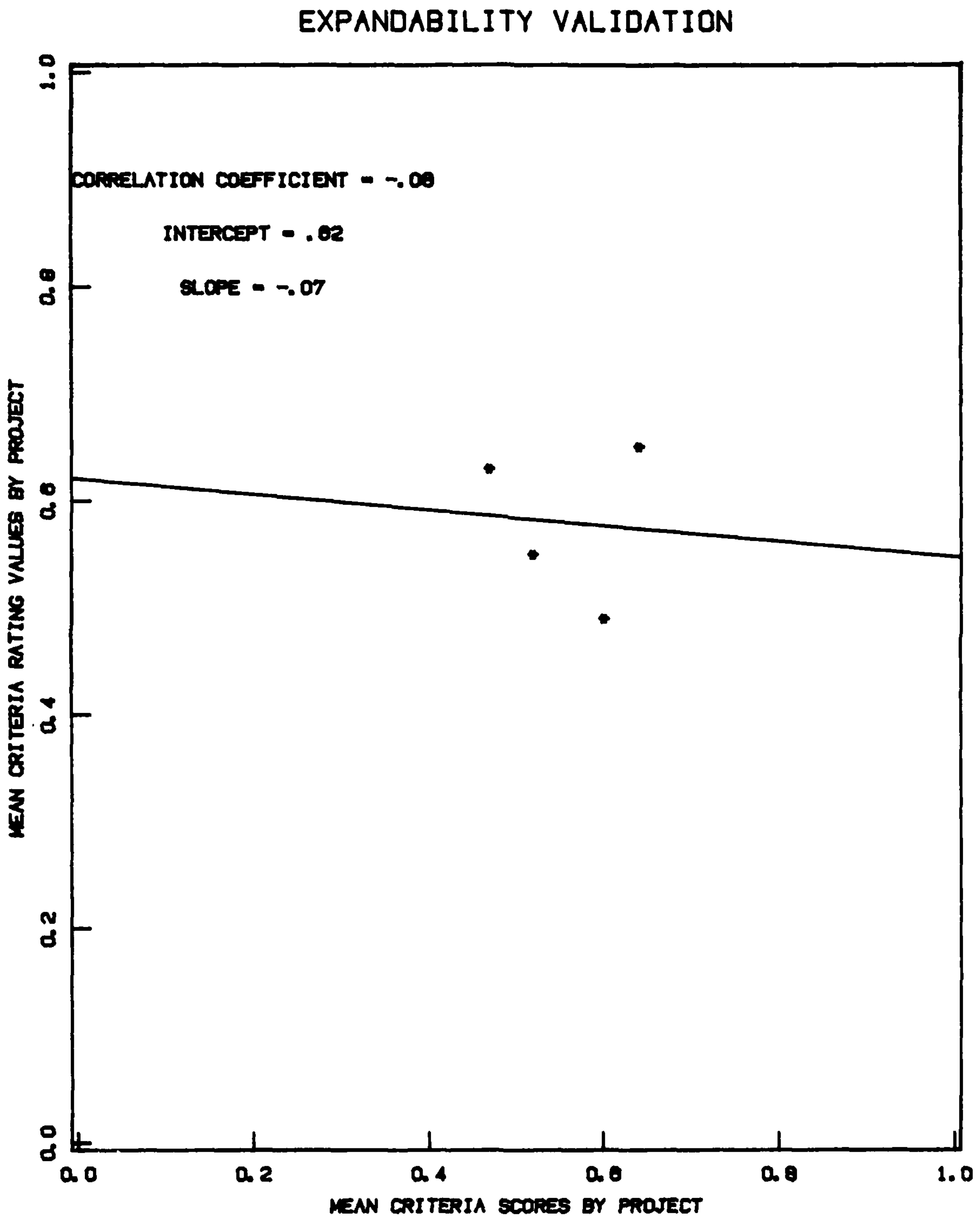


Figure 5.4-2 Expandability Mean Criteria Scores vs. Mean Criteria Rating Values

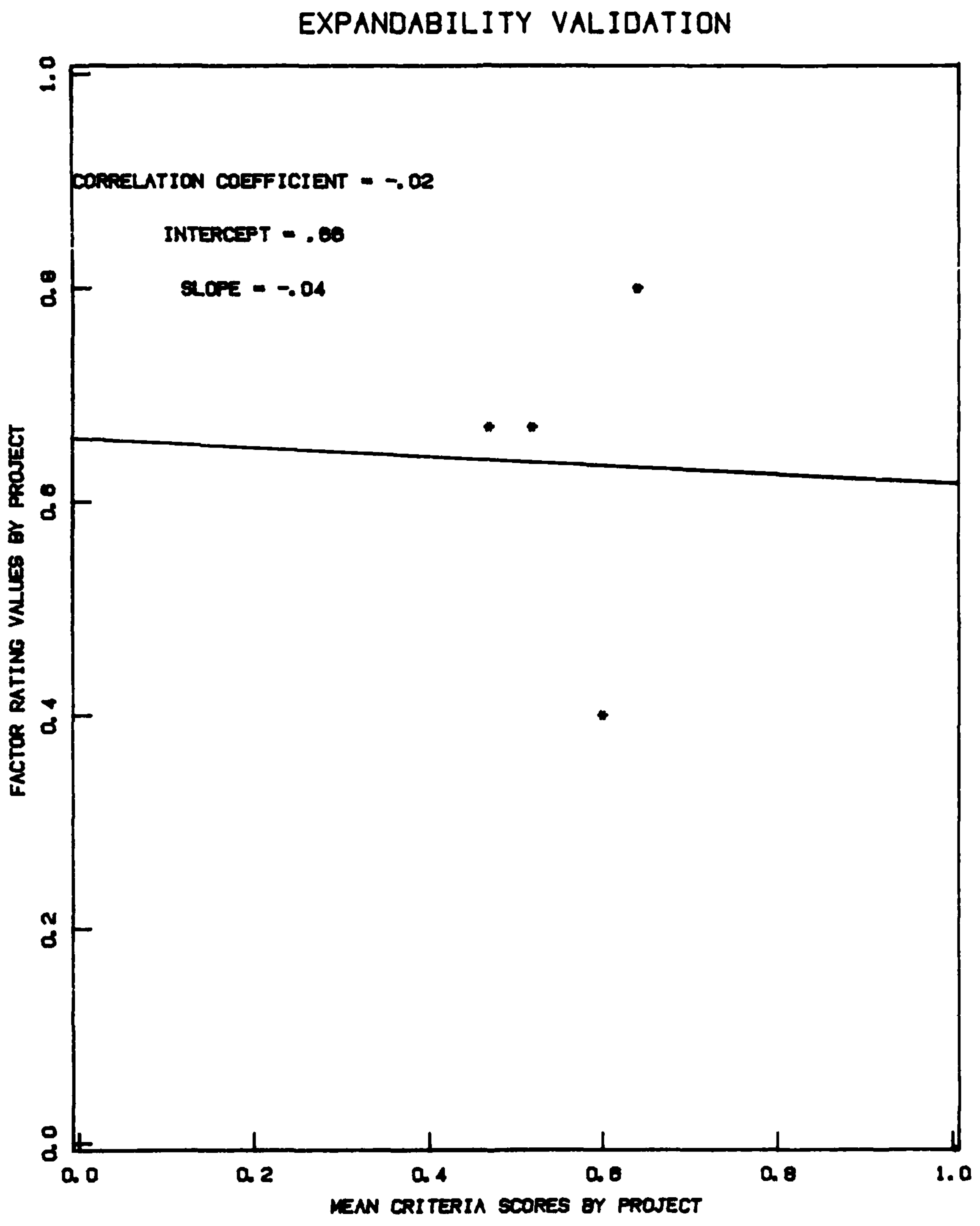


Figure 5.4-3 Expandability Mean Criteria Scores vs. Factor Rating Values

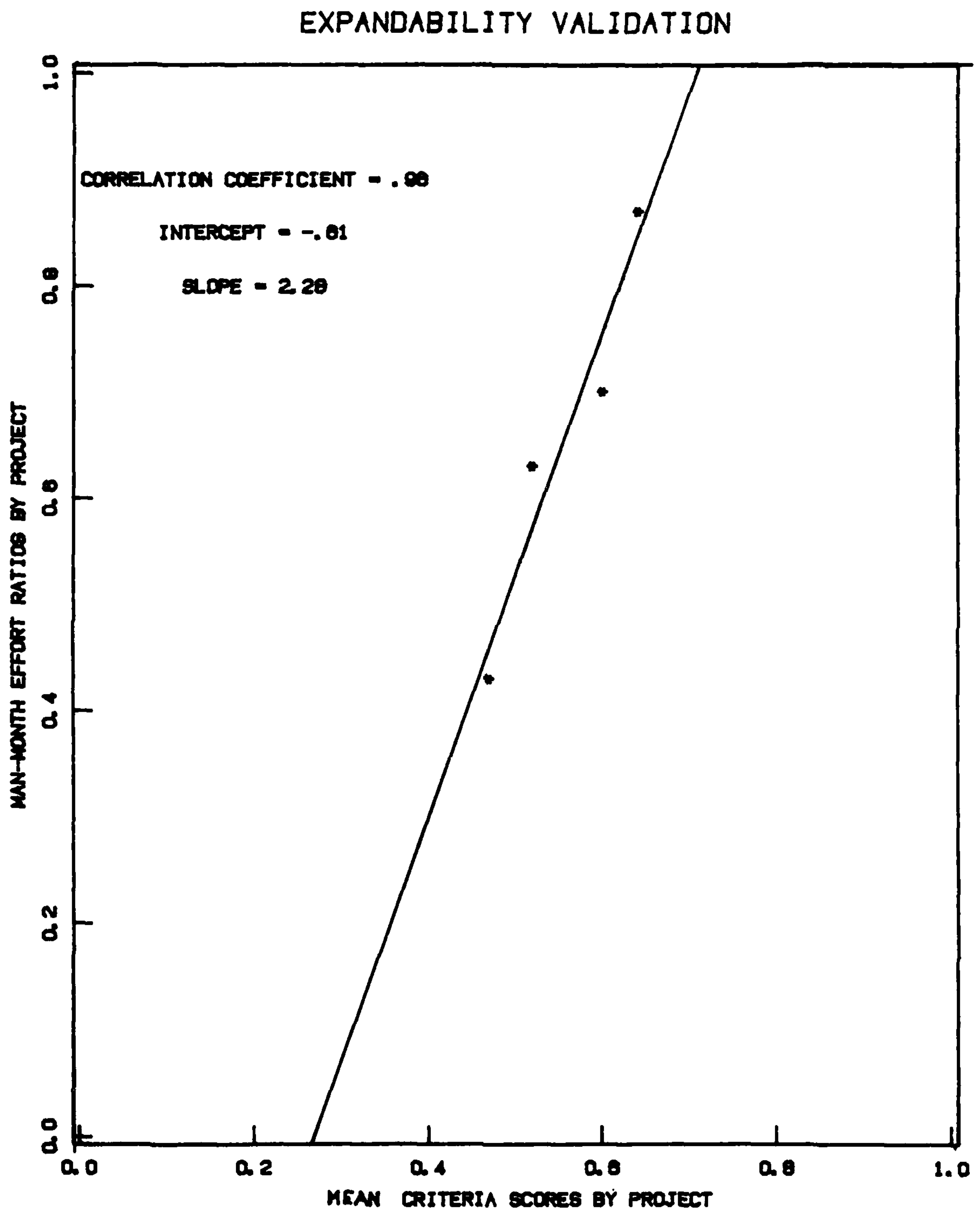


Figure 5.4-4 Expandability Mean Criteria Scores vs. Man-month Ratios

5.4.2 Survivability Data Analysis

The data collected from the metric worksheets for survivability realted metrics, as described in section 5.2.1, are summarized in Table 5.4-4. This table contains the average score of all metric scores for each criterion of survivability, for each of the four projects. The mean score for survivability is computed as the average of the criteria scores for each project.

Table 5.4-4 SURVIVABILITY CRITERION SCORES

CRITERIA		PROJECT			
		A	B	C	D
ANOMALY MAN.	AM	.66	.75	.60	.48
AUTONOMY	AU	.72	.61	.84	.51
DISTRIBUTEDNESS	DI	.84	.68	1.00	.94
MODULARITY	MO	.45	.36	.44	.28
RECONFIGURAB.	RE	.54	.71	1.00	.88
MEAN CRITERIA SCORE		.64	.62	.78	.62

The data collected from the rating questionnaires described in section 5.2.2 are summarized in Table 5.4-5. This table contains the quasi-delphi quality rating value for each criterion of survivability for each of the four projects. The survivability quality rating is then computed as the average of the criteria ratings (Mean Criteria Rating) for each project. The survivability Factor Rating for each project is the quasi-delphi quality rating value for the attribute survivability, taken from the rating questionnaires.

Table 5.4-5 SURVIVABILITY QUALITY RATING VALUES

CRITERIA		PROJECT A	B	C	D
ANOMALY MAN.	AM	1.00	.93	.80	.50
AUTONOMY	AU	.83	.47	.30	.50
DISTRIBUTEDNESS	DI	.70	.63	.85	.70
MODULARITY	MO	.50	.80	.65	.67
RECONFIGURAB.	RE	.93	.73	.95	.70
MEAN CRITERIA RATING		.79	.71	.71	.61
FACTOR RATING		.93	.93	.95	.68

The data from Tables 5.4-4 and 5.4-5 was statistically analyzed to compute regression equations and correlation coefficients to compare survivability scores (metrics) with survivability rating values (quality levels). The results of this analysis are shown in Table 5.4-6.

Table 5.4-6 Summary of Statistical Analysis for Survivability

STATISTIC	QUALITY RATING Individual Criteria	Mean Criteria	Factor
Intercept	.59	.61	.37
Slope	.18	.14	.76
Correlation Coefficient	.21	.15	.45

$Y = a + bX$

a = Intercept	Y = Quality Rating
b = Slope	X = Criteria Score (Metrics)

The data in Tables 5.4-4 and 5.4-5 and the regression/correlation results in Table 5.4-6 are plotted in Figures 5.4-5 through 5.4-7. The X axis represents the survivability criteria scores (calculated from the metrics) and the Y axis represents the various survivability quality ratings. All of these correlations were positive, but were fairly low (.15 to .45), with the highest correlation at the factor rating level.

A second set of correlation studies were then conducted using metric data from Worksheets 1 and 2 only since there is a greater amount of metric data for survivability on the system level worksheets. For example, 100% (7 of 7) of the worksheet entries for the criterion reconfigurability (metric RE.1) are on worksheets 1 and 2; 87% (13 of 15) of the worksheet entries for the criterion distributedness (metric DI.1) are on worksheets 1 and 2. The concern for survivable software is primarily affected by architecture rather than details at the code level. Survivability is affected by such things as how processes and functions are distributed among the nodes, how information/data is distributed throughout the system, and the scheme for communicating among nodes. Survivability is not very dependent on details such as the types of constructs used in the code or the number of comments on listings. (The correlations using scores from worksheets 1 and 2 only were not performed for expandability as this quality factor is dependent both on top-level architectures and on details at the code level.)

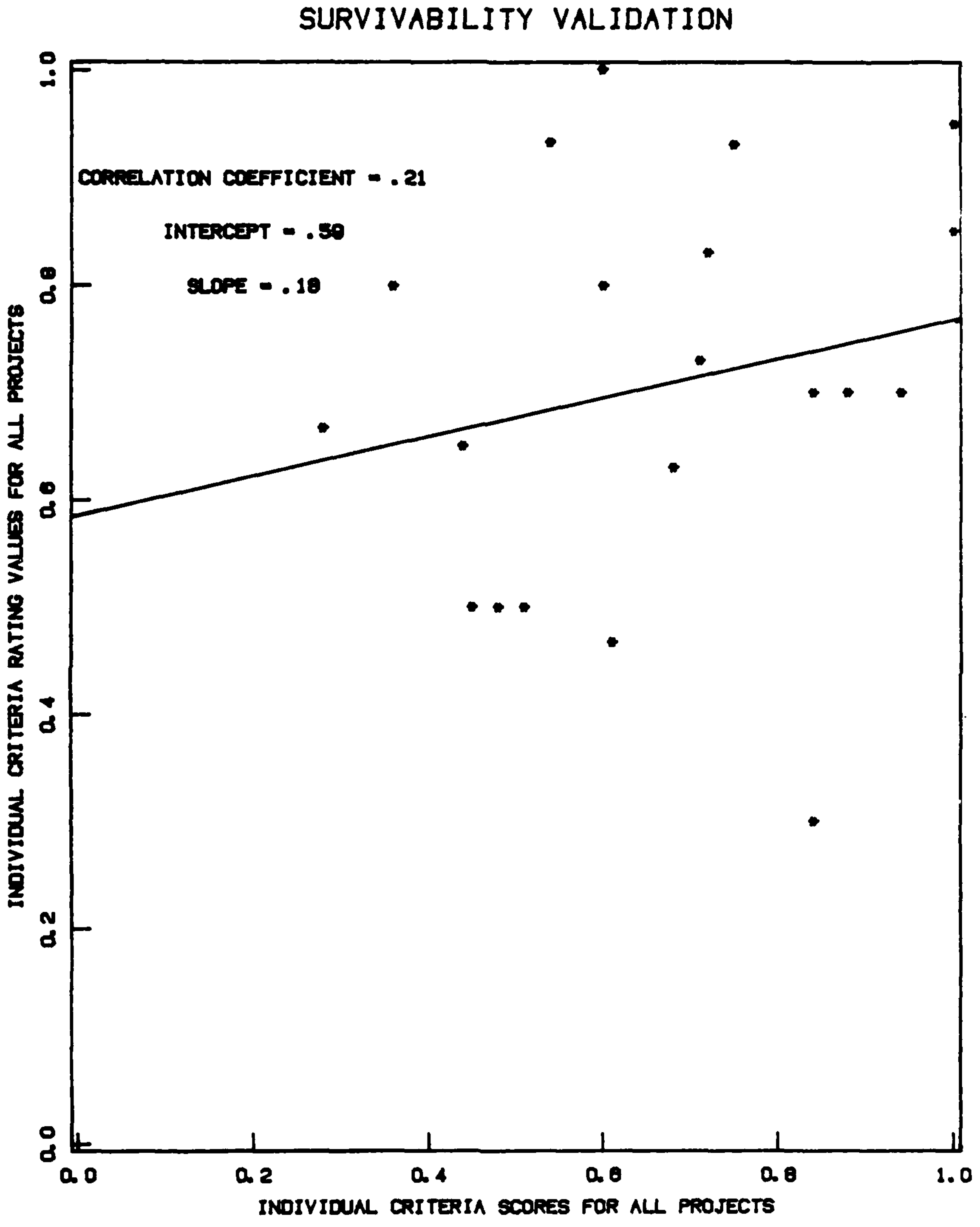


Figure 5.4-5 Survivability Individual Criteria Scores for All Projects vs. Individual Criteria Rating Values for All Projects

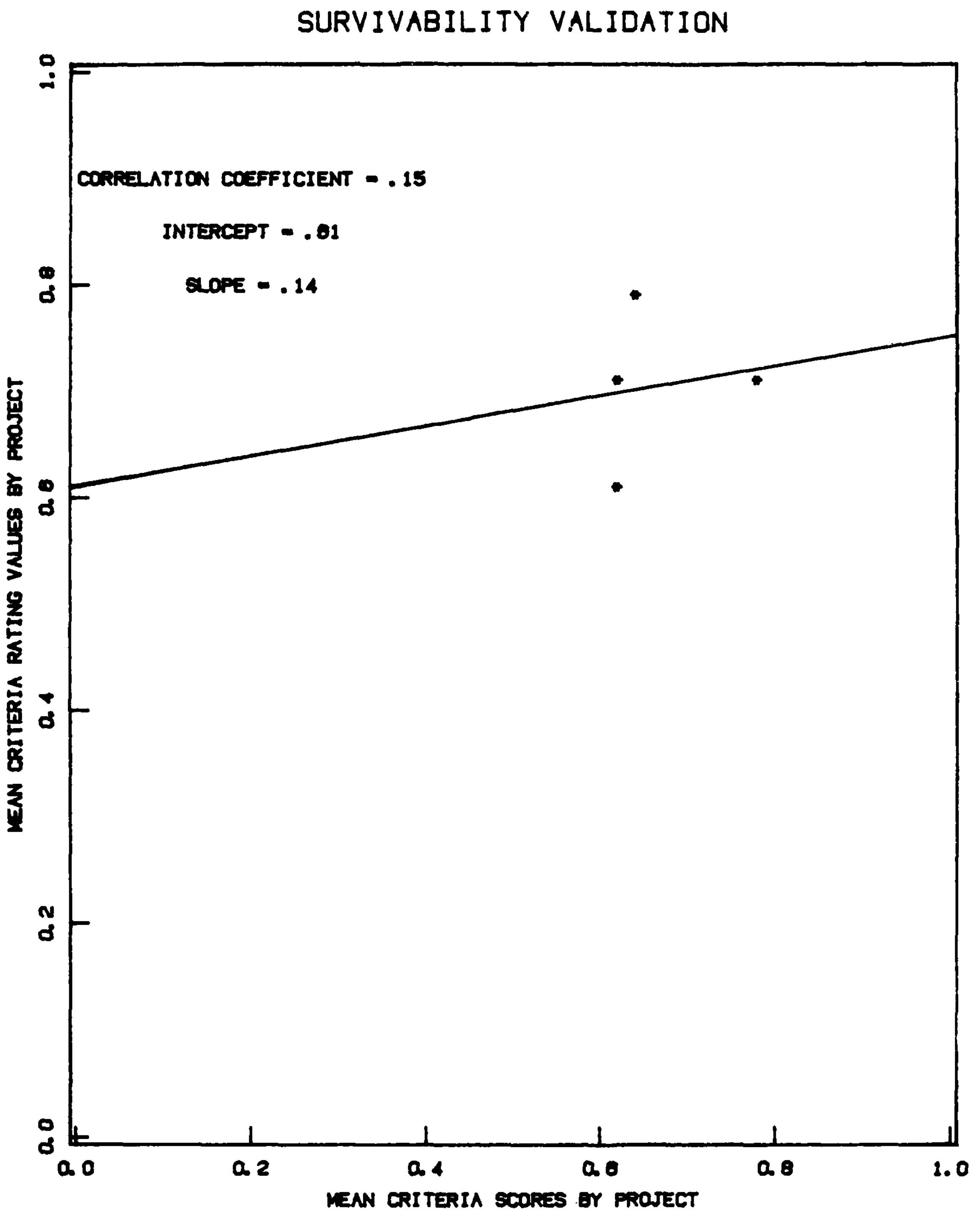


Figure 5.4-6 Survivability Mean Criteria Scores vs. Mean Criteria Rating Values

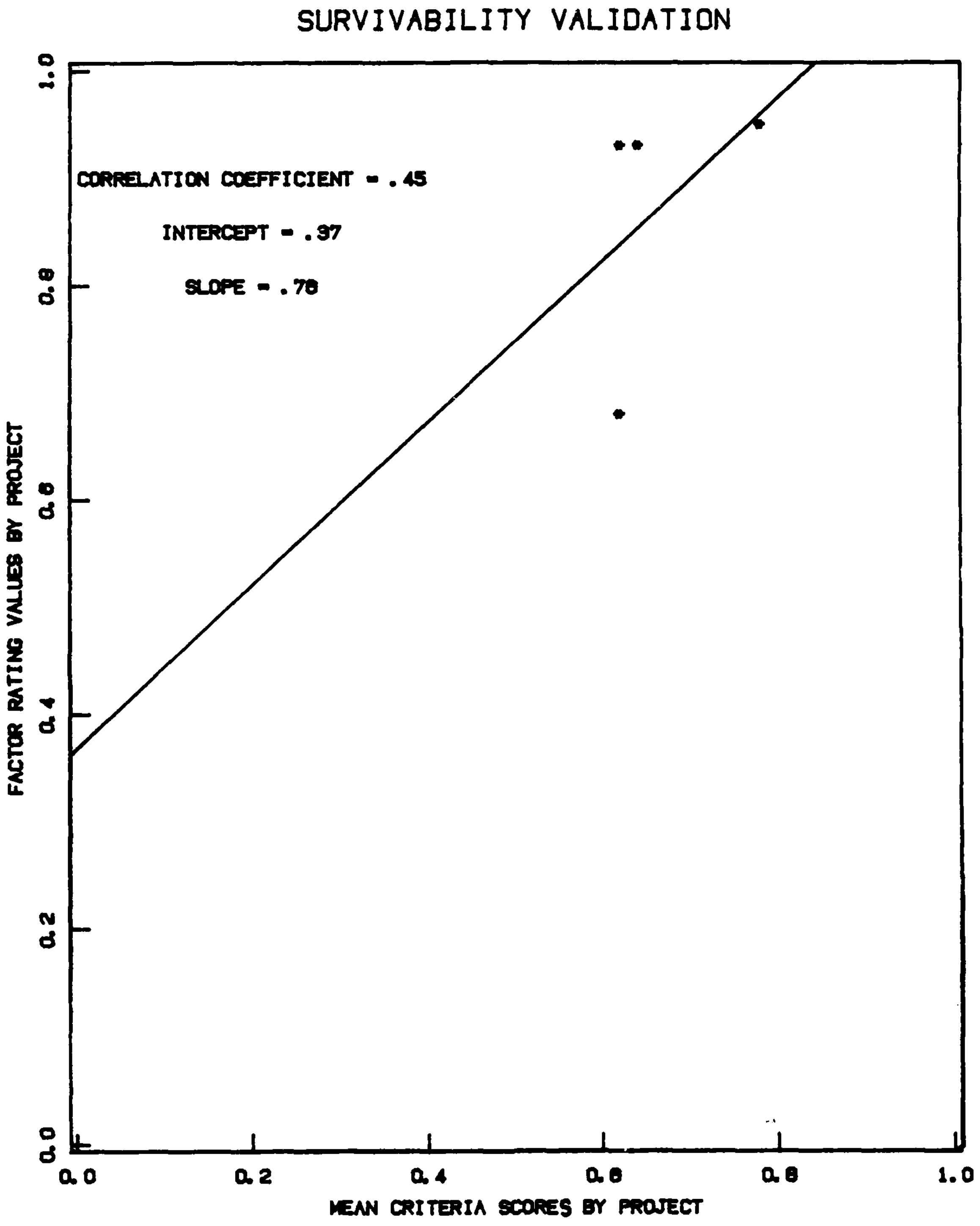


Figure 5.4-7 Survivability Mean Criteria Scores vs. Factor Rating Values

The data summarized from only worksheets 1 and 2 is shown in Table 5.4-7.

Table 5.4-7 Survivability Criteria Scores, Worksheets 1 and 2

	PROJECT			
	A	B	C	D
MEAN CRITERIA SCORES	.68	.66	.73	.62

The results of the statistical analysis of data from Table 5.4-7 and 5.4-5 is shown in Table 5.4-8.

Table 5.4-8 Summary of Statistical Analysis for Survivability, Worksheets 1 and 2

STATISTIC	QUALITY RATING	
	Mean Criteria	Factor
Intercept	.08	-.66
Slope	.93	2.27
Correlation Coefficient	.58	.81

$Y = a + bX$

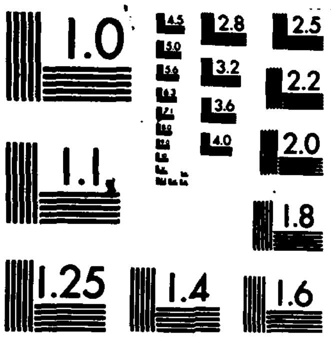
a = Intercept	Y = Quality Rating
b = Slope	X = Mean Criteria Score (Metrics)

The correlation coefficient, using only worksheets 1 and 2 metric data, are significantly higher than using data from all worksheets when evaluating survivability. Again the highest correlation was at the factor rating level (.81).

The data in Tables 5.4-7 and 5.4-5 and the regression/correlation results [illegible] plotted in Figures 5.4-8 and 5.4-9.

AD-A137 955 SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS 2/2
VOLUME 1(U) BOEING AEROSPACE CO SEATTLE WA
T P BOWEN ET AL. JUL 83 RADC-TR-83-175-VOL-1
UNCLASSIFIED F30602-80-C-0330 F/G 9/2 NL

END
FILMED
3-84
DTIC


1.0
4.5
5.0
5.6
6.3
7.1
8.0
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

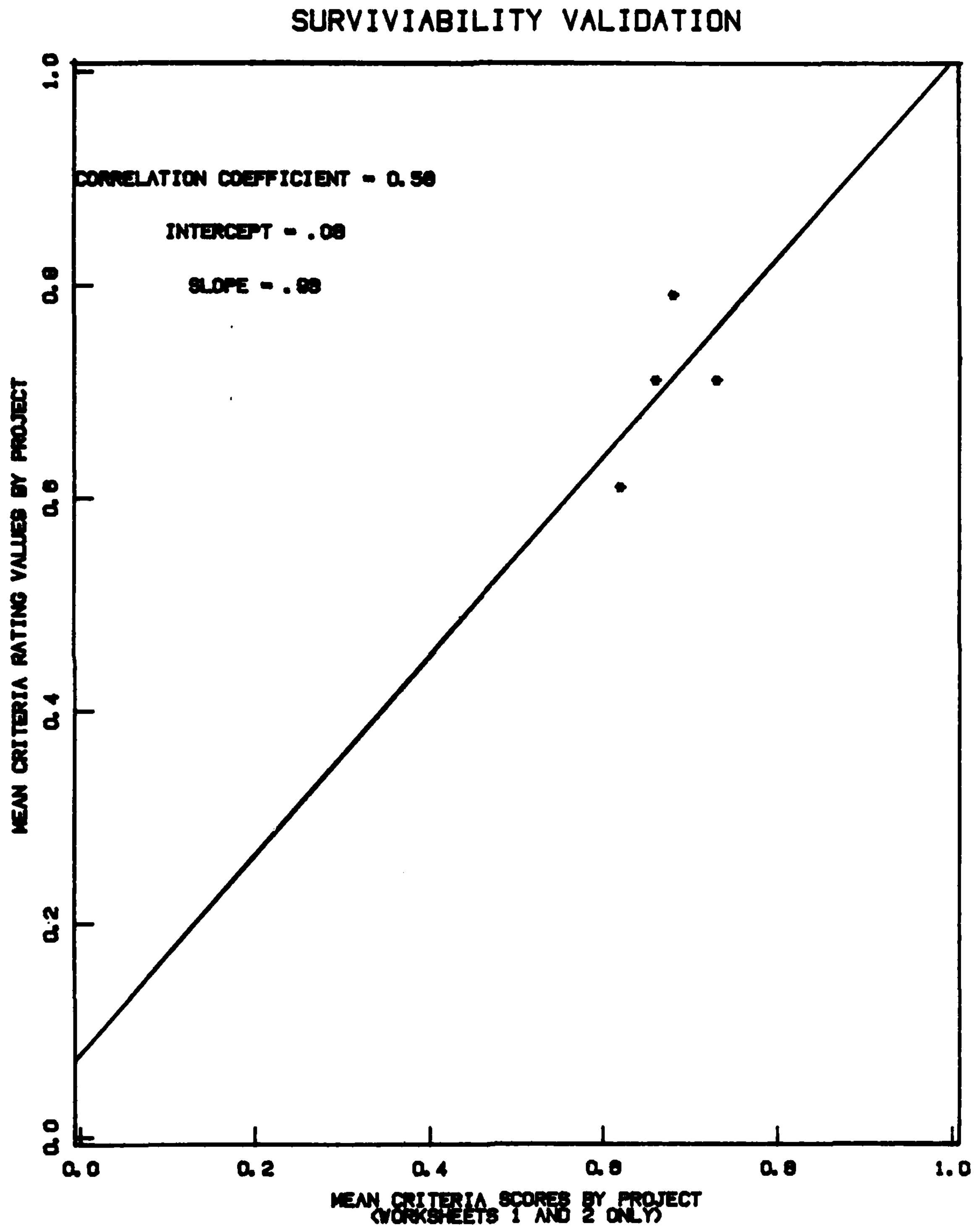


Figure 5.4.8 Survivability Mean Criteria Scores vs. Mean Criteria Rating Values

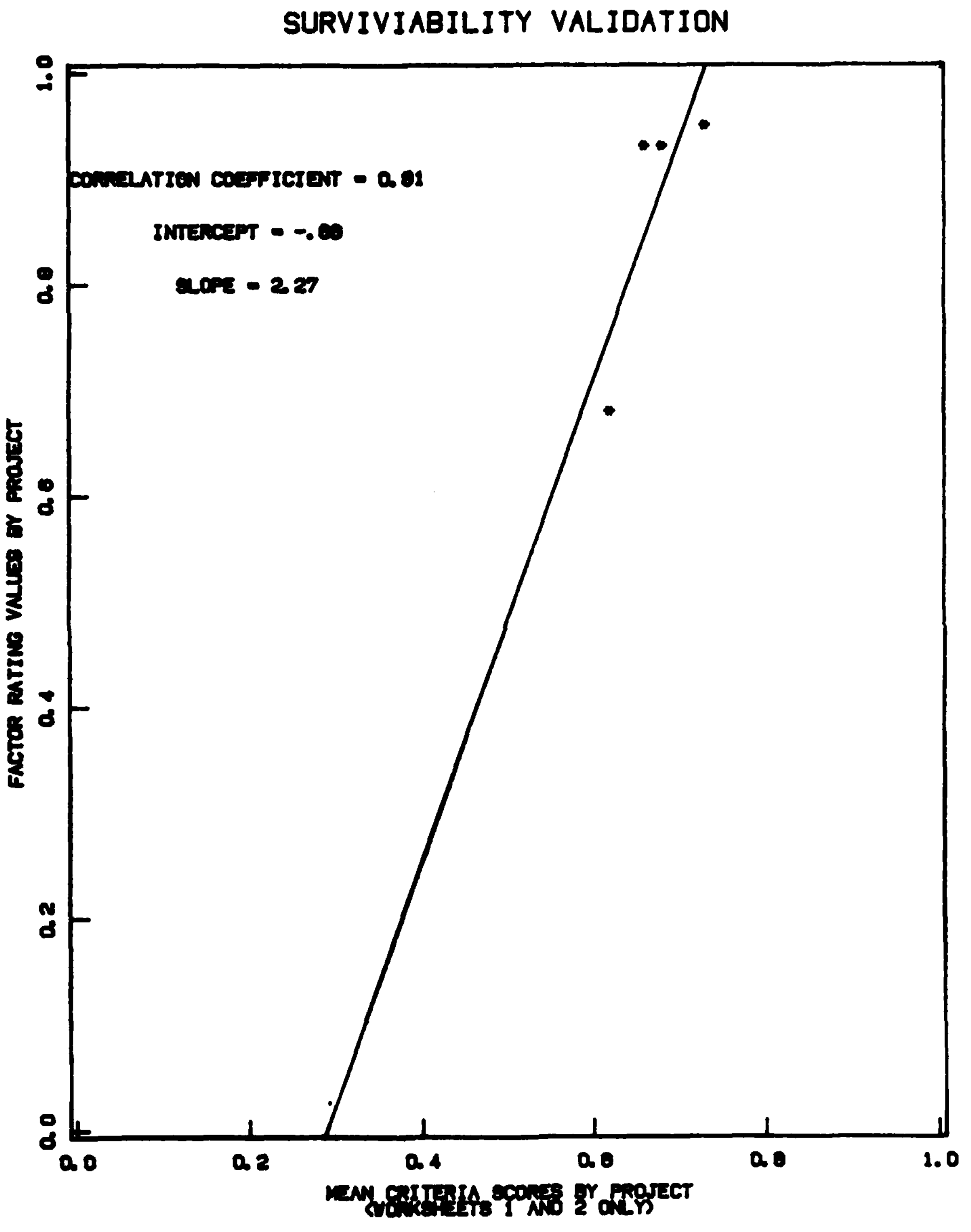


Figure 5.4-9 Survivability Mean Criteria Scores vs. Factor Rating Values

5.5 CONCLUSIONS

5.5.1 Expandability Conclusions

The statistical analyses of expandability (metric) scores and expandability quality rating values support the addition of the quality factor expandability (and its criteria and metrics) to the software quality framework. The mean criteria scores (average metric scores) are highly correlated with the quality rating values of the Effort Rating (R_E), as shown in Figure 5.4-4. For this plot, the mean criteria scores (M_E) from Table 5.4-1 were used for the x-axis, and the quality rating values based on the Effort Rating (R_E) from Table 5.4-2 were used for the y-axis. (The Effort Rating is calculated from man-month effort estimates on the rating questionnaires.) The best regression fit to the data is:

$$R_E = -0.61 + 2.28\ M_E$$

The correlation coefficient calculated for this regression is 0.96, significantly above random. This correlation provides a high confident level for the addition of the factor expandability.

(Note that the y-intercept is not equal to zero. Therefore, an attempt to find a best fit to this data, while forcing the line through the origin (with y-intercept equal to zero) would yield higher residual error than this fit.)

A second conclusion that can be drawn from the statistical analyses of expandability is that the Effort Rating, based on man-month effort estimation, is probably more useful for validation than the Mean Criteria Rating or Factor Rating, both based on an estimation of attributes of the software as judged by an experienced observer. The Effort Rating yielded the highest correlation (.96) with mean criteria scores (metric scores). The correlations for the Mean Criteria Rating and the Factor Rating were near zero. This indicates that metric scores and these rating values are essentially uncorrelated— little better than random. Expandability validation plots and a summary of the analysis are found in section 5.4.1.

5.5.2 Survivability Conclusions

The statistical analyses of survivability (metric) scores and survivability quality rating values provide a high confidence level for the addition of the quality factor survivability (and its criteria and metrics) to the software quality framework.

As shown in section 5.4.2, all correlation coefficients attained when using data from all four worksheets in validating survivability are positive with a range of .15 to .45. In comparison, the range of the correlation coefficients attained when using data from only worksheets 1 and 2 is from .58 to .81. This indicates that the data analysis involving worksheets 1 and 2 supports the rationale for pursuing this approach.

However, researchers on this contract feel that the quasi-delphi questionnaire approach is useful, but not conclusive. An improved method of estimating survivability ratings (perhaps one which includes probability data or effort ratios in man-months) could improve the confidence in the validation of survivability.

SECTION 6.0
RECOMMENDATIONS FOR FUTURE RESEARCH

6.1 QUALITY FACTORS AND CRITERIA

6.1.1 Ultrastability

A suggested new criterion is ultrastability. This criterion relates to "the capacity of a system to withstand perturbations which have not been forseen by the designer." (BEE 75, p.108) This criterion, by its very definition, cannot be evaluated by the system designer until the design process is complete, and the maintenance process initiated. It would fall within the factors of survivability, reliability, and maintainability.

6.1.2 Prototypeability

A suggested new criterion is prototypeability. Prototypeability measures the extent to which a system design can be developed through a sequence of executable prototypes possessing increasing functionality, with more and more implementation detail and alternative designs being the rationale for the successive prototype systems. The question is whether or not the design can evolve with the implementation, and whether or not the design as implemented can support design decisions. The prototype approach is a special case within the distributed system life cycle, where the system passes several times through each life cycle phase.

Advantages of Prototypeability

Design can evolve, in ways not limited by initial requirements. Competing implementation approaches are repeatedly examined. It is easy to generalize requirements and meet the newly generalized requirements in a new prototype system. It takes less time to backtrack to an earlier phase of the life cycle. Initial prototype systems become early starting points for maintenance and enhancement processes.

Disadvantages of Prototypeability

There may be language or system changes which require additional training or personnel. Since the design is not fixed, it is hard to implement and administer the system efficiently. Separate languages for prototypes and production versions may require a software translation phase in the life cycle. There is likely to be a shift from an interpreter-based approach in prototypes to a faster compiler-based approach for production. It is difficult to avoid changes and interruptions, since the prototype approach is advertised to adapt to these so quickly.

6.1.3 Distributedness, Integration, and Transparency

Fusi and Sommi of IBM Italy (FUS 81) draw distinctions useful to our definition of distributedness, and to our discussions of evolveability and the distributed system life cycle.

> "to use the term 'computer network' as a synonym for distributed system is quite misleading. In fact, a computer network provides nothing more than the basic communication between computers, which is a necessary but not a sufficient ingredient to create a distributed environment.... When processors communicate using a shared storage and some special instructions, the system is a multi-processor system. When memory is not shared and the communication between processors is obtained by using interconnection media like, for example, I/O channels, coaxial cables, optical fibers, teleprocessing links, satellite links, public data networks, the system may more properly be defined as a Distributed Data Processing (DDP) system. If we define 'integration' the capability of accessing a set of interconnected data processing elements as a single system, a DDP system offers a lower degree of integration than a multi-processor system, and more or less presents the same problems as far as distribution aspects, but (a) it is a system which covers a greater generality of aspects of distribution (both local and geographically dispersed processors to be interconnected) and (b) it offers the possibility to move smoothly towards processor integration. This means that a DDP system may evolve to a fully distributed system (complete integration) following the evolution of the communications media (higher bandwidth etc.) and the evolution of user applications."

They also define "Transparency" for a distributed system of virtual machines:

(a) the ability to utilize the same programming interface both for local and remote communications

(b) the possibility to map remote communications on different transmission subsystems without user involvement

6.1.4 Parallelism vs. Provability

Hoare and Chen (HOA 81) introduce a programming notation to describe the behavior of groups of parallel processes communicating with each other over a network of named channels. Denotational and axiomatic definitions are aimed towards a method for proving the correctness of parallel programs. This ambitious attempt allows some non-deterministic program assertions to be proved, but fails to show whether or not the program will freeze into a deadlock state. It seems clear, therefore, that automated methods for the design of distributed systems cannot yet be expected to validate the correctness of particular parallel programs. Thus, for distributed systems, testing and validation can be much more difficult than for sequential systems whose behavior may be mathematically modelled in terms of denotational semantics.

6.1.5 Responsiveness vs. Consistency

Kamoun, Kleinrock, and Muntz (KAM 81) discuss integrity and consistency issues for distributed databases. They present a mathematical analysis of Le Lann's ticketing method of concurrency control for fully redundant multiple copy distributed database systems, which is being implemented in the Sirius-Delta Project at INRIA (Institut National de Recherche en Informatique et Automatique, France). The scheme requires updates at each site to be processed in the order of origin, as opposed to the order of arrival. The characteristics of variable communication delays in the network affect performance. One goal of this analysis is "that some analytical quantitative tools become available to help identify performance characteristics of the various schemes" for concurrency control. It is perhaps too early to define criteria or metrics for this, but a comprehensive survey of concurrency control algorithms may be found in (BER 80) (Bernstein,P.A., and Goodman,N. "Fundamental Algorithms for Concurrency Control in Distributed Database Systems", Tech.Rept. CCA-80-05, Computer Corporation of America, 15 Feb 1980) One scheme for concurrency control is of particular relevance to distributed computer systems with high survivability requirements. This, the virtual ring architecture design, is described in Volume III.

6.2 CENTROID OF A DISTRIBUTED SYSTEM

There is a gradation between highly centralized and highly peripheralized distributed systems. There is also a continuum of different configurations of dispersion in a geographically dispersed distributed system. A novel means of representing this is the Centroid of Computation. A related concept is the Computational Moment of Inertia, which may be proposed as a step towards a quantitative measure of distributedness. The following discussion leads up to formal definition of these constructs.

As Metcalfe and Boggs explain (see reference below):

> "One can characterize distributed computing as a spectrum of activities varying in their degree of decentralization with one extreme being remote computer networking and the other extreme being multiprocessing. Remote computer networking is the loose interconnection of previously isolated widely separated and previously monolithic and serial computing systems from increasingly numerous and smaller pieces computing in parallel. Near the middle of this spectrum is local networking the interconnection of computers to gain the resource sharing of computer networking and the parallelism of multiprocessing".

First, consider the special case of a distributed system which consists of precisely two processor nodes connected by a straight-line communications link. If the two nodes have identical performance, we may model the system behavior as a spacial average, consisting of one processor with twice the performance of the two original node processors, considered to be located halfway between them, in the center of the communications link. Similarly, four identical processors located at the corners of a square of links may be modeled in some aspects as one processor of four times the unit performance, located at the center of the square. Of course, this neglects the very distributedness of the system. Any information regarding communication delays between processors has been lost, as has information relating to real-time asymmetries in processing.

If two non-identical processors are located along a communications link which is straight and of unit length, we may locate the single Virtual processor at a point between the two whose distance is a weighted measure of the relative performances. For example, a processor of unit power and a processor of nine-times-unit power connected by a link of ten units length may be modeled as a virtual processor one unit from the larger processor. From this location, the products of distance and power in each direction are equal.

The determination of the centroid of computation, given a planar distribution of processors, is formally the same as determining the center of mass, where mass plays the role of computational power. One benefit of this is to indicate where, geographically, the "effective" center is of a distributed system.

This model may be extended to model the degree of geographical centralization or decentralization. The procedure is to, first, locate the centroid of computation. Next, take the square-root of the sum of the squares of the products of distance from centroid of computation to each processor and the power of that processor. This number will be smaller when the more powerful processors are near the centroid of computation, and larger when the more powerful processors lie at the extremities of the distributed system. This comparison remains valid between two distributed systems of the same maximum distance. This metric is analogous to the rotational moment-of-inertia.

These ideas need to be strengthened and formalized to be of greater applicability to the study of distributed computer systems. One problem with the above analysis is that the DISTANCE between two computers is not the sole measure of their "effective" distance. It would be valuable to include the TIME it takes to communicate between two processors, and the WIDTH of the channel in bits-per-second. We therefore, define the VIRTUAL DISTANCE (VD) between two processors:

> VIRTUAL DISTANCE between processor I and processor J = VD(I,J) = the TIME it takes to transmit one bit from I to J, in the shortest possible path = (communications delay from I to J) / (channel width in bits).

Actually, taking the speed of light into account, we can correct for very long geographic distances as well. The following formula shows the relative contributions of inherent communications delay and transmission delay. For speed-of-light, of course, one may substitute the appropriate propagation velocity if lower.

$$VD = \left(\frac{(\text{geographic distance})^2}{(\text{speed-of-light})^2} + (\text{time-to-communicate-1-bit})^2 \right)^{½}$$

Metcalfe and Boggs ("Ethernet Distributed Packet Switching for Local Computer Networks", Robert M. Metcalfe and David R. Boggs, Xerox Palo Alto Research Center, Communications of the ACM, July 1976, Vol 19, No. 7, p. 395) consider a related measure

> "The product of separation and bit rate, now about 1 gigabit-meter per second (1 Gbmps), is an indication of the limit of current communications technology and can be expected to increase with time":

Activity	Separation	Bit Rate
Remote networks	10 km	0.1 Mbps
Local networks	10 - 0.1 km	0.1 - 10 Mbps
Multiprocessors	0.1 km	10 Mbps

We can also define computing power in terms of COMPUTATIONAL MASS. The COMPUTATIONAL MASS (CM) of a computer is the number of bits processed per second. More exactly,

COMPUTATIONAL MASS of PROCESSOR I = CM(I) =
(Operations-per-second) X (Number-of-bits-processed-per-operation)

(i.e. 1 Megaflop with 32-bit words means CM = 32,000,000)

The measurement of "operations-per-second" is difficult itself, as it can be made only with respect to some benchmark software which drives the system through some distribution of operations. It is difficult, usually, to make such comparisons between processors manufactured by different vendors. In this discussion, we assume that such a benchmark environment exists, and that different processing powers may be compared.

Continuing our analogy with the elementary physics of kinematics, we substitute VIRTUAL DISTANCE for distance and COMPUTATIONAL MASS for mass in the conventional formulae for centroid, moment-of-inertia, etc. in physics.

VIRTUAL DISTANCE is not quite like geographical distance, which forms a euclidean metric space. VD satisfies 3 of the 4 axioms of a metric space:

VD(i,i) = 0
VD(i,j) = 0 i=j
VD(i,k) equal.or.less.than VD(i,j)+VD(j,k) triangle inequality
but VD(i,j) does not necessarily equal VD(j,i) consider half-duplex

Because Virtual Distance is not symmetrical (VD(i =/= VD(j,i)) the representation of the distributed system in terms of nodes and links technically a "directed graph" with weighted vertices (nodes) and arcs (links).

The VIRTUAL CENTER of a distributed system, not necessarily the same as its CENTROID, is the processor with the minimum distance to all other processors.

CENTER(i) if.and.only.if (for.all j not.equal.to i) (there.exists k such.that)
(VD(i,k) less.than VD(j,k))

If we wish to add more processing power to some existing processor in a distributed system, for example, it makes sense to add it to the CENTER processor so as to minimize the Virtual Distance to all other processors which may need to indirectly access that power. We can then define the RADIUS of the systems:

RADIUS = Maximum (over j) of VD(i,j) where CENTER(i) = True

The Radius is therefore one measure of distributedness. But it is not yet what we want. The reason is: such a measure of distributedness only takes into account the "worst case" of the greatest Virtual Distance that can be found between any pair of processors. What is missing here is representation of how closely the typical Virtual Distance between processors approaches the worst case. Clearly, there is a difference between a system in which ALL processors are far apart, and a system in which almost all processors are very close except for one remote station. We need a formula which considers all links and all nodes, to be able to distinguish between this and other cases.

We can define our measure of Distributedness in terms of the "computational moment of inertia of the system with respect to the CENTER:

$$CI = \sum_{j=1}^{N} VD(i, j)^2 \times CM(j)$$ where CENTER(i) = True and N = number of nodes

What we have now is a single number which sums the contributions of all the processors. This function is linear with respect to the Computational Mass (power) of the processors, and non-linear with respect to the Virtual Distances from the Center (with an emphasis on the most distant processors). The function is zero for a uniprocessor, and larger as the Virtual Distances between processors in a distributed system increases. It is therefore useful as a measure of distributedness.

There are a number of additional aspects of distributedness which need to be incorporated into this model. For example, the interface with human users. What is the computational power of human operator in such a system? What is the virtual distance between a person and the various types of terminals?

When processors, sensors, or communications links are moving (as with space-, air-, or sea-borne embedded systems) how do these equations change with time? How does the measure of distributedness change in a system whose processors appear (when the system is expanded) or disappear (hardware failure) or change their connectivity (reconfigurability)?

6.3 REFERENCES FOR FUTURE RESEARCH

Recent references in the literature of Distributed Systems which emphasize system evaluation methodology, and which may be further examined in the next phases of this research include:

(A & J) Anderson, George A. and Jensen, E. Douglas, "Computer Interconnection Structures: Taxonomy, Characteristics, and Examples", ACM Computing Survey, Dec. 1975

(BAL) Balkovich, Edward E., "On the performance of decentralized software", PER EVAL REV 9:173-80, Summer 1980. Reports on performance measures for decentralized software written in a programming language for distributed computer systems. 16 references.

(BIE) "A Performance Tool for Design and Installation Support of Distributed Database Systems", J.Bieber & S.Florek, DCS 440-447

(BNL) Locanthi, B.N., "The Homogeneous Machine", Technical Report 3759, Computer Science Department, California Institute of Technology, Pasadena, CA, Jun. 1980

(CHW) (Chu, W. and Chew, P., "Computer Networks: A Tutorial", IEEE Comput. Soc., New York, 1980)

(COD) (COD 82) Codd, J.F. et. al., "Data Base Debate", Computerworld, Sep. 1982, p. 14

(COF) Coffman, E.G., Gelenbe, E., & Plateau,B., "Optimization of the number of copies in a distributed data base system", PER EVAL REV 9:257-63, Summer 1980. Considers the effect on system performance of the distribution of a data base in the form of multiple copies at distict sites. 4 references.

(DCS) (DCS = Proc. 1st International Conference on Distributed Computing Systems, Huntsville, Alabama, 1-5 Oct 1979, IEEE 79CH1445-6 C)

(DCS2) (Proc.2nd International Conf. on Distributed Computing Systems, Paris, France, 8-10 April 1981, Computer Society Press, IEEE No.81CH1591-7)

(DSI 82) Strategy for a DOD Software Initiative, Dr. Edith Martin, Deputy Undersecretary of Defense for Research and Engineering (Research and Advanced Technology), 1 October 1982

(GAU) Gausnell, William A., "Optimizing a distributed processing system", SMALL SYS 8:20-3, Oct 1980. Recommends ways to work towards an optimum system based on previous experience, and on methods currently being used at Bell Labs.

(GOK) Goke, R. and Lipovski, G.J., "Banyan Networks for Partitioning on Multiprocessor Systems", Proc. 1st Ann. Symp. Computer Architecture, 1973, pp. 21-30

(HOA) Hoare, C.A.R. and Chen, Z.C., "Partial Correctness of Communicating Sequential Processes", DCS2, 1-12

(LAR) (Larson, R., "Tutorial: Distributed Control", IEEE Comput. Soc., New York, 1979)

(KAM) Kamoun, F., Kleinrock, L., and Muntz, R., "Queueing Analysis of the Ordering Issue in a Distributed Database Concurrency Control Mechanism", DCS2, p.13-23

(LES 80) Lesser, V., "Working Papers in Distributed Computation—I Cooperative Distributed Problem Solving", Computer and Information Science Department, University of Massachusetts at Amherst, Amherst, MA 01003

(MAM) Mamrak, Sandra A. "Sizing distributed systems: Overview and Recommendations", NTIS, may 1980, 25pp, PB80-184377. $5.00. Presents an overview of sizing techniques,a brief discussion of the factors that affect choosing one or a combination of techniques, and a set of recommendations for choosing tools for sizing distributed systems.

(MAU) (Mauchley, John; Personal communication to J. Post, Philadelphia, PA, June 1978)

(MCG) (McGlynn, D. R., "Distributed Processing and Data Communications", Wiley-Interscience, New York, 1978)

(MIT) "Evaluating the Trade-off Between Centralized and Distributed Computing", I.Mitrani & K.C.Sevcik, DCS 520-527

(PRE) Preparata, F.P., and Vuillemin, J., "The Cube-Connected Cycles: A Versatile Network for Parallel Computation" Comm. ACM, Vol. 24, No. 5, May 1981, pp. 300-306)

(RAM) Ramamoorthy, C.V., "The design methodology of distributed computing systems", NTIS, May 1980, 101 pp., AD-A086 690/5. $9.00. Develops performance evaluation techniques for asynchronous concurrent systems using the Petri net approach. Analysis techniques for deadlocks in asynchronous concurrent systems are explored. A need for adaptive reconfiguration techniques is established along with necessary and sufficient conditions for reconfigurability

(ROT) Rothnie, J., Bernstein, P., and Shipman, D., "Tutorial: Distributed Database Management", IEEE Comput. Soc., Los Alamedos, California, 1978

(SCH) Scherr, A.L., "Distributed Data Processing", IBM Syst. J., Vol. 17, No. 4, 1978, p.338

(SEG) "A Majority Consensus Algorithm for the Consistency of Duplicated and Distributed Information", J.Seguin, G.Sergeant, P.Wilms, DCS 617-624

(SIE) Siegel, M.J., "A Model of SIMD Machines and a Comparison of Various Interconnection Networks", IEEE Trans-Computers, Vol. C-28, No. 12, Dec. 1979, p. 907-917

(SMI 77) Smith, R. A., "The Contract Net: A Formalism for the Control of Distributed Problem Solving", 5th International Joint Converence on Artificial Intelligence, Cambridge, Massachusetts, 1977

(SPE 81) (Sperry, Dr. Roger; personal communication to J. Post, Caltech, 1973)

(STN) Stone, H., "Parallel Processing with the Perfect Shuffle", IEEE Trans. Computers, Vol. C-20, No. 2, Feb. 1971, pp. 153-161

(STZ) Seitz, C., and Hewitt, C., personal communication cited in "A Survey of Highly Parallel Computing", IEEE Computer, Jan 1980, ref. 60, p.23

(SUL) Sullivan, Kenneth M., "Does distributed processing pay off?", DATAMATION 26:192-6, Sep 1980. Compares the costs involved in running a job on a large scale centalized mainframe to the costs of running the same job in a minicomputer dedicated to that task in a distributed processing environment.

(SVO) Svobodova, Liba "Performance problems in distributed systems." INFOR 18:21-40, Feb 1980. Examines the performance issues involved in the decision to use a distributed system, performance problems that arise in operating such a system, and the characteristics and implementation of the needed performance evaluation tools. 40 references.

(SYK) Sykes, David J., "The economics of distributed systems", COMPCON p.8-15, Fall 80 Discusses computer price/performance data communications, systems availability, and database replication. A hierarchically distributed transaction processing system and an equivalent centralized system are compared. 10 references.

(TAN) Tanenbaum, A. S., "Computer Networks", Prentice-Hall, Englewood Cliffs, New Jersey, 1981

(THE) (Theirauf, R. J., "Distributed Processing Systems", Prentice-Hall, Englewood Cliffs, New Jersey, 1978)

(THU) (Thurber, K., "Tutorial: A Pragmatic View of Distributed Processing Systems", IEEE Comput. Soc., Los Alemedas, California, 1980)

(WU) Wu, Chuan-lin, & Feng, Tsu-yun. "A software technique for enhancing performance of a distributed computing system", COMPSAC p.274-80, 1980. Demonstrates a software technique to effectively match the task execution and a distributed architecture. 17 references.

APPENDIX A
BIBLIOGRAPHY

(The contents of this appendix can be found in Volume III).

APPENDIX B
GLOSSARY OF KEY TERMS

(The contents of this appendix can be found in Volume III).

APPENDIX C
VALIDATION WORKSHEETS

METRIC WORKSHEET 1	SYSTEM:	DATE:	
REQUIREMENTS ANALYSIS/SYSTEM LEVEL	NAME:	INSPECTOR:	

1.1 STRUCTURE (RELIABILITY, MAINTAINABILITY, VERIFIABILITY, FLEXIBILITY, REUSABILITY EXPANDABILITY, SURVIVABILITY, PORTABILITY, INTEROPERABILITY)

1.*	Is an organization of the system/network provided which identifies all software functions and functional interfaces in the system? DI.1(1)	Y	N
2.*	Number of major functions. SI.1(2)		
3.*	Are there no duplicate functions? SI.1(2)	Y	N
4.*	Is there a definitive statement of the requirements for the distibution of information within the data base? DI.1(3)	Y	N
5.*	Is there an organization of the data base provided which identifies the types of system-level information and the information flow within the system? DI.1(2)	Y	N
6.*	Is there a definitive statement of requirements for code to be written according to a programming standard? SI.4(17)	Y	N
7.*	Is there a definitive statement of requirements for processes, functions, and modules to have loose coupling? MO.3(1)	Y	N
8.*	Is there a definitive statement of requirements for processes, functions, and modules to have high cohesion? MO.3(2)	Y	N

1.2 TOLERANCE (RELIABILITY, SURVIVABILITY)

3.	Are there definitive statements of the error tolerance of input data? AM.2(1)	Y	N
4.	Are there definitive statements of the requirements for recovery from computational failures? AM.3(1)	Y	N
5.	Is there a definitive statement of the requirement for recovery from hardware faults? AM.4(1)	Y	N
6.	Is there a definitive statement of the requirements for recovery from device errors? AM.5(1)	Y	N
7.*	Are there definitive statements of the requirements for recovery from communication errors? AM.6(1)	Y	N
8.*	Are there definitive statements of the requirements for system recovery from node or communication failures? AM.7(1)	Y	N

* = New, ** = Modified

METRIC WORKSHEET 1	SYSTEM:	DATE:
REQUIREMENTS ANALYSIS/SYSTEM LEVEL	NAME:	INSPECTOR:

1.6 CHANGEABILITY (FLEXIBILITY, EXPANDABILITY)

1.*	Is there a definitive statement of requirements for spare storage capacity (memory and auxiliary storage)? AG.1(2,3)	Y	N
2.*	Is there a definitive statement of requirements for spare processing capacity? AG.2(3)	Y	N
3.*	Is there a definitive statement of requirements for spare I/O and communication channel capacity? AG.3(1,2)	Y	N
4.*	Is there a definitive statement of requirements for interface compatibility among all the processors, communication links, memory devices, and peripherals? AG.4(1)	Y	N
5.*	Is there a specific requirement for providing performance/price information for enhancement trades? AG.4(2)	Y	N
6.*	Do specifications identify new technology tradeoff areas for software? AG.4(3)	Y	N
7.*	Do software specifications include requirements for the criteria of the quality factor expandability? AG.4(4)	Y	N

1.7 SYSTEM INTERFACES (INTEROPERABILITY, SURVIVABILITY)

6.*	Are processes and functions separated as logical "wholes" to minimize interface complexity? AU.1(1)	Y	N
7.*	Are there specific requirements for each CPU/system to have a separate power source? AU.2(1)	Y	N
8.*	Are there specific requirements for each software scheduling unit to test its own operation, communication links, memories, and peripherals? AU.2(3)	Y	N
9.*	Are there specific requirements for the software system to include a word processing capability? AU.2(3)	Y	N
10.*	Are there specific requirements for network communication capabilities in the event of failure of a node or communication link? RE.1(1)	Y	N
11.*	Are there specific requirements for a node to rejoin the network when it has been recovered? RE.1(4)	Y	N

METRIC WORKSHEET 1	SYSTEM:	DATE:
REQUIREMENTS ANALYSIS/SYSTEM LEVEL	NAME:	INSPECTOR:

1.8 DATA BASE (SURVIVABILITY, USABILITY, INTEGRITY, EXPANDABILITY, CORRECTNES RELIABILITY, MAINTAINABILITY)

1.*	Is there a definitive statement of the requirements for maintaining data base integrity under anomalous conditions? RE.1(2)	Y	N
2.*	Are there specific requirements for file/library accessibility from each node? DI.1(4)	Y	N
3.*	Are there specific requirements for a virtual storage structure? VR.1(1)	Y	N

1.10 INSPECTOR'S COMMENTS

Make any general or specific comments that relate to the quality observed while applying th checklist.

METRIC WORKSHEET 2	SYSTEM:	DATE:
DESIGN/SYSTEM LEVEL	NAME:	INSPECTOR:

2.1 STRUCTURE (RELIABILITY, MAINTAINABILITY, VERIFIABILITY, FLEXIBILITY, REUSABILIT EXPANDABILITY, SURVIVABILITY, PORTABILITY, INTEROPERABILITY, INTEGRITY, USABILITY)

1.*	Is an organization of the system provided which identifies all functions and functional interfaces in the system? DI.1(1)	Y	N
2.	Is a hierarchy of system identifying all modules in the system provided? SI.1(1)	Y	N
3.	Are there no duplicate functions? SI.1(2)	Y	N
4.*	Is an organization of the data base provided which identifies all functional groupings of data and data flow within the system? DI.1(2)	Y	N
5.*	Are there provisions for selecting alternate processing capabilities? DI.1(5)	Y	N
6.*	Are critical system functions distributed over redundant elements or nodes? DI.1(6)	Y	N
7.*	Does the distribution of control functions ensure network operation/integrity under anomalous conditions? DI.1(7)	Y	N
8.*	Are logical structure and function separated in the design? DI.1(8)	Y	N
9.*	Are physical structure and function separated in the design? DI.1(9)	Y	N
10.*	Number of nodes that can be removed and still have each node able to communicate with each remaining node: DI.1(10)		
11.*	Do processes and functions have loose coupling? MO.3(1)	Y	N
12.*	What is the cohesion value of processes and functions? MO.3(2)		
13.*	Can each user utilize the system as though it were dedicated to that user? VR.1(4)	Y	N
14.*	Is the user presented with a complete logical system without regard to physical topology? VR.1(5)	Y	N
15.*	Do module descriptions include identification of module interfaces? SI.1(10)	Y	N

2.2 TOLERANCE (RELIABILITY, SURVIVABILITY)

3.	Is concurrent processing centrally controlled? AM.1(1)	Y	N
4.*	Is parallel processing centrally controlled? AM.1(4)	Y	N
5.	How many error conditions are reported by the system? AM.1(2)		

* = New, ** = Modified

METRIC WORKSHEET 2	SYSTEM:	DATE:	
DESIGN/SYSTEM LEVEL	NAME:	INSPECTOR:	

6.	How many of those errors are automatically fixed or bypassed and processing continues? AM.1(2)		
7.	How many, require operator intervention? AM.1(2)		
8.	Are there provisions for recovery from hardware faults? AM.4(2)	Y	N
9.	Are there provisions for recovery from device errors? AM.5(2)	Y	N
10.*	Are there provisions for recovery from communication errors? AM.6(2)	Y	N
11.*	Are there provisions for system recovery from node or communication failures? AM.7(2)	Y	N

2.6 CHANGEABILITY (FLEXIBILITY, REUSABILITY, EXPANDABILITY)

1.*	Percent of memory capacity uncommitted. AG.1(2)		
2.*	Percent of auxiliary storage capacity uncommitted. AG.1(3)		
3.*	Percent of speed capacity uncommitted. AG.2(3)		
4.*	Spare I/O channel capacity. AG.3(1)		
5.*	Spare communication channel capacity. AG.3(2)		
6.*	Are processors, communication links, memory devices, and peripherals compatible (of a common vendor or model)? AG.4(1)	Y	N
7.*	Does documentation reveal performance/price of software/system for enhancement trades? AG.4(2)	Y	N
8.*	Do specifications identify new technology tradeoff areas for software? AG.4(3)	Y	N
9.*	Do software specifications include requirements for the criteria of the quality factor expandability. AG.4(4)	Y	N
10.	Number of modules. GE.1(1)		
11.	Based on hierarchy or a call/called matrix, how many modules are called by more than one module? GE.1(1)		

2.7 SYSTEM INTERFACES (INTEROPERABILITY, SURVIVABILITY)

10.*	Is configuration of communication links such that failure of one node/link will not disable communication among other nodes? RE.1(1)	Y	N
11.*	Can node rejoin the network when it has been recovered? RE.1(4)	Y	N
12.*	Is data replicated at two or more distinct nodes? RE.1(5)	Y	N

METRIC WORKSHEET 2	SYSTEM:	DATE:
DESIGN/SYSTEM LEVEL	NAME:	INSPECTOR:

13.*	Are processes and functions separated as logical "wholes" to minimize interface complexity? AU.1(1)	Y N
14.*	Estimated number of lines of interface code. AU.1(2)	
15.*	Estimated number of interface modules. AU.1(3)	
16.*	Estimated time engaged in communication. AU.1(4).	
17.*	Does each CPU/system have a separate power source? AU.2(1)	Y N
18.*	Does each scheduling unit test its own operation, communication links, memories, and peripherals? AU.2(2)	Y N
19.*	Does the software system include a word-processing capability? AU.2(3)	Y N

2.8 DATA BASE (RELIABILITY, MAINTAINABILITY, VERIFIABILITY, FLEXIBILITY, REUSABILITY EXPANDABILITY, USABILITY, INTEGRITY, SURVIVABILITY, CORRECTNESS)

1.	Number of unique data items in data base SI.1(6)	
2.	Number of preset data items SI.1(6)	
3.	Number of major segments (files) in data base SI.1(7)	
4.*	Is the data base structured so that at least one copy of a file/library resides at a node which is accessible to all other nodes? DI.1(4)	Y N
5.*	Is the data base structured so that users need not care about changes in the actual storage structure of data? VR.1(2)	Y N
6.*	Are there provisions for maintaining data base integrity under anomalous conditions? RE.1(3)	Y N
7.*	Can users manipulate data as if it were not replicated elsewhere in the system? VR.1(3)	Y N

2.11 INSPECTOR'S COMMENTS

Make any general or specific comments about the quality observed while applying this checklist.

METRIC WORKSHEET 3	SYSTEM NAME:	DATE:
DESIGN/MODULE LEVEL	MODULE NAME:	INSPECTOR:

3.1 STRUCTURE (RELIABILITY, MAINTAINABILITY, VERIFIABILITY, FLEXIBILITY, REUSABILITY, EXPANDABILITY, CORRECTNESS, PORTABILITY, INTEROPERABILITY, SURVIVABILITY)

1.*	Is an organization of the system provided which identifies all modules and module interfaces? DI.1(1)	Y	N
2.*	Is an organization of the data base provided which identifies all data base modules and module interfaces? DI.1(2)	Y	N
3.	How many Decision Points are there? SI.3		
4.	How many subdecision Points are there? SI.3		
5.	How many conditional branches are there? SI.3		
6.	How many unconditional branches are there? SI.3		
7.	Is the module dependent on the source of the input or the destination of the output? SI.1(3)	Y	N
8.	Is the module dependent on knowledge of prior processing SI.1(3)	Y	N
9.	Number of entrances into modules SI.1(5)		
10.	Number of exits from module SI.1(5)		
11.*	Does the module description include input, output, processing, and limitations? SI.1(4)	Y	N
12.*	Is code written according to a programming standard? SI.4(17)	Y	N
13.*	Are macros and subroutines used to avoid repeated and redundant code? SI.4(18)	Y	N
14.*	Number of input parameters. SP.1(1)		
15.*	Number of output values used. SP.1(2)		
16.*	Number of output parameters. SP.1(2)		
17.*	Can the same function not be accomplished by multiple variant forms? SP.1(3)	Y	N
18.*	Does each function and module have loose coupling? MO.3(1)	Y	N
19.*	What is the cohesion value of each function and module? MO.3(2)		
20.*	Do module descriptions include identification of module interfaces? SI.1(10)	Y	N

3.2 TOLERANCE (RELIABILITY, SURVIVABILITY)

1.	When an error condition is detected, is it passed to calling module? AM.1(3)	Y	N
3.	Are values of inputs range tested? AM.2(2)	Y	N
4.	Are conflicting requests and illegal combinations identified and checked? AM.2(3)	Y	N

* = New, ** = Modified

METRIC WORKSHEET 3	SYSTEM NAME:	DATE:
DESIGN/MODULE LEVEL	MODULE NAME:	INSPECTOR:

5.	Is there a check to see if all necessary data is available before processing begins? AM.2(5)	Y N
6.	Is all input checked, reporting all errors, before processing begins? AM.2(4)	Y N
7.	Are loop and multiple transfer index parameters range tested before use? AM.3(2)	Y N
8.	Are subscripts range tested before use? AM.3(3)	Y N
9.	Are outputs checked for reasonableness before processing continues? AM.3(4)	Y N
10.*	Are checksums computed and transmitted with all messages? AM.6(3)	Y N
11.*	Are checksums computed and compared upon message reception? AM.6(4)	Y N
12.*	Are the number of transmission retries limited? AM.6(5)	Y N
13.*	Are adjacent nodes checked periodically for operational status? AM.7(3)	Y N
14.*	Are there alternate strategies for message routing? AM.7(4)	Y N

3.5 REFERENCES (MAINTAINABILITY, FLEXIBILITY, VERIFIABILITY, PORTABILITY, REUSABILITY, INTEROPERABILITY, EXPANDABILITY, SURVIVABILITY)

5.	Number of calling sequence parameters MO.2(3)	
6.	How many calling sequence parameters are control variables? MO.2(3)	
7.	Is input passed as calling sequence parameters MO.2(4)	Y N
8.	Is output passed back to calling module? MO.2(5)	Y N
9.	Is control returned to calling module? MO.2(6)	Y N
10.	Is temporary storage shared with other modules? MO.2(7)	Y N

3.6 CHANGEABILITY (FLEXIBILITY, REUSABILITY, EXPANDABILITY)

1.	Is logical processing independent of storage specification? AG.1(1)	Y N
2.*	Percent of memory allocation uncommitted. AG.1(2)	
3.	Are accuracy, convergence, or timing attributes and limitations parametric? AG.2(1)	Y N
4.	Is module table driven? AG.2(2)	Y N
5.*	Percent of cycle time allocation uncommitted. AG.2(3)	
6.*	I/O channel time allocation uncommitted. AG.3(1)	
7.*	Communication channel time allocation uncommitted. AG.3(2)	
8.	Does the module not mix input, output and processing functions in same module? GE.2(1)	Y N
9.	Number of machine dependent functions performed? GE.2(2)	

METRIC WORKSHEET 3	SYSTEM NAME:	DATE:
DESIGN/MODULE LEVEL	MODULE NAME:	INSPECTOR:

10. Is processing not data volume limited? GE.2(3)	Y N
11. Is processing not data value limited? GE.2(4)	Y N

3.7 SYSTEM INTERFACES (SURVIVABILITY)

1.* Estimated lines of interface code. AU.1(2)	
2.* Estimated lines of source code. AU.1(2)	
3.* Estimated number of interface modules. AU.1(3)	
4.* Estimated time engaged in communication. AU.1(4)	

3.10 INSPECTOR'S COMMENTS

Make any specific or general comments about the quality observed while applying this checklist.

METRIC WORKSHEET 4	SYSTEM NAME:	DATE:
SOURCE CODE/MODULE LEVEL	MODULE NAME:	INSPECTOR:

4.1 STRUCTURE (RELIABILITY, MAINTAINABILITY, VERIFIABILITY, FLEXIBILITY, PORTABILIT REUSABILITY, INTEROPERABILITY, EXPANDABILITY, SURVIVABILITY)

1.	Number of lines of code MO.2(2)		
2.	Number of lines excluding comments SI.4(2)		
4.	Number of declarative statements SI.4(9)		
5.	Number of data manipulation statements SI.4(9)		
6.	Number of statement labels (Do not count format statements SI.4(6)		
7.	Number of entrances into module SI.1(5)		
8.	Number of exits from module SI.1(5)		
9.	Maximum nesting level SI.4(7)		
10.	Number of decision points (IF, WHILE, REPEAT, DO, CASE) SI.3		
11.	Number of sub-decision points. SI.3		
12.	Number of conditional branches (computed go to) SI.4(8)		
13.	Number of unconditional branches (GOTO, ESCAPE) SI.4(8)		
14.	Number of loops (WHILE, DO) SI.4(3)		
15.	Number of loops with jumps out of loop SI.4(3)		
16.	Number of loop indices that are modified SI.4(4)		
17.	Number of constructs that perform module modifications (SWITCH, ALTER) SI.4(5)		
18.	Number of negative or complicated compound boolean expressions SI.4(2)		
19.	Is a structured language used SI.2	Y	N
20.	Is flow top to bottom (are there no backward branching GOTOs) SI.4(1)	Y	N
21.*	Is code written according to a programming standard? SI.4(17)	Y	N
22.*	Are macros and subroutines used to avoid repeated and redundant code? SI.4(18)	Y	N

4.2 TOLERANCE (RELIABILITY, SURVIVABILITY)

1.	How many loop and multiple transfer index parameters are not range tested before use? AM.3(2)		
2.	Are subscript values range tested before use? AM.3(3)	Y	N
3.	When an error condition occurs, is it passed to the calling module? AM.1(3)	Y	N

METRIC WORKSHEET 4	SYSTEM NAME:	DATE:
SOURCE CODE/MODULE LEVEL	MODULE NAME:	INSPECTOR:

4.	Are the results of a computation checked before outputting or before processing continues? AM.3(4)	Y N
5.*	Are all data available prior to processing? AM.2(5)	Y N

4.5 REFERENCES (MAINTAINABILITY, VERIFIABILITY, FLEXIBILITY, PORTABILITY REUSABILIT INTEROPERABILITY, EXPANDABILITY, SURVIVABILITY)

1.	Number of calls to other modules MO.2(1)	
3.	Number of calling sequence parameters MO.2(3)	
4.	How many elements in calling sequences are not parameters? MO.2(3)	
5.	How many of the calling parameters (input) are control variables? MO.2(3)	
6.	How many parameters passed to or from other modules are not defined in this module? MO.2(3)	
7.	Is input data passed as parameter? MO.2(4)	Y N
8.	Is output data passed back to calling module? MO.2(5)	Y N
9.	Is control returned to calling module? MO.2(6)	Y N

4.6 CHANGEABILITY (FLEXIBILITY, REUSABILITY, EXPANDABILITY)

1.	Is module table driven? AG.2(2)	Y N
2.	Are there any limits to data values that can be processed? GE.2(4)	Y N
3.	Are there any limits to amounts of data that can be processed? GE.2(3)	Y N
4.	Are accuracy, convergence and timing attributes parametric? AG.2(1)	Y N
5.*	Amount of memory used. AG.1(2)	

4.7 INPUT/OUTPUT (RELIABILITY, PORTABILITY, REUSABILITY, SURVIVABILITY)

4.	Are inputs range-tested (for inputs via calling sequences, global data, and input statements) AM.2(2)	Y N
5.	Are possible conflicts or illegal combinations in inputs checked? AM.2(3)	Y N
6.	Is there a check to determine if all data is available prior to processing? AM.2(5)	Y N

* = New, ** = Modified

METRIC WORKSHEET 4	SYSTEM NAME:	DATE:
SOURCE CODE/MODULE LEVEL	MODULE NAME:	INSPECTOR:

		Y	N
7.	Is all input checked, reporting all errors, before processing begins? AM.2(4)		
8.*	Number of lines of interface code. AU.1(2)		
9.*	Number of modules with interface code. AU.1(3)		

4.9 DATA (CORRECTNESS, RELIABILITY, MAINTAINABILITY, VERIFIABILITY, EFFICIENCY FLEXIBILITY, REUSABILITY, EXPANDABILITY)

1.	Number of local variables SI.4(10)	
2.	Number of global variables SI.4(10)	
3.	Number of global variables renamed EF.4(3)	
6.*	Number of executable statements. SI.4(11)	

4.11 DYNAMIC MEASUREMENTS (EFFICIENCY, RELIABILITY, FLEXIBILITY, EXPANDABILITY SURVIVABILITY)

2.	During module/development testing, what was run time? AG.2(3)	
6.*	Amount of I/O channel capacity used. AG.3(1)	
7.*	Amount of communication channel capacity used. AG.3(2)	
8.*	Time engaged in communication. AU.1(4)	

4.12 INSPECTORS COMMENTS

Make any general or specific comments that relate to the quality observed while applying this checklist.

APPENDIX D
RATING QUESTIONNAIRES

PROJECT ________________

SURVIVABILITY

On a scale of 1 to 10 (ten highest) please rate the software for this system with respect to the following attributes.

ATTRIBUTE	DESCRIPTION	RATING
o Anomaly Management	o Those attributes of the software which provide for continuity of operations under and recovery from non-nominal conditions.	________
o Automony	o Those attributes of the software which determine its dependency on interfaces.	________
o Distributedness	o Those attributes of the software which determine the degree to which software functions are geographically or logically separated within the system.	________
o Modularity	o Those attributes of the software which provide a structure of highly cohesive modules with optimum coupling.	________
o Reconfigurability	o Those attributes of the software which provide for continuity of system operation when one or more processors, storage units, or communications links fail.	________
o Survivability	o Probability that software will continue to perform or support critical functions when a portion of the system is inoperable.	________
o Augmentability	o Those attributes of the software which provide for expansion of capability for functions and data.	________
o Generality	o Those attributes of the software which provide breadth to the functions performed with respect to the application.	________

PROJECT ______________

EXPANDABILITY

On a scale of 1 to 10 (ten highest) please rate the software for this system with respect to the following attributes.

ATTRIBUTE	DESCRIPTION	RATING
o Modularity	o Those attributes of the software which provide a structure of highly cohesive modules with optimum coupling.	______
o Simplicity	o Those attributes of the software which provide for the definition and implementation of functions in the most non-complex and understandable manner.	______
o Specificity	o Those attributes of the software which provide singularity in the definition and implementation of functions.	______
o Virtuality	o Those attributes of the software which present a system that does not require user knowledge of the physical, logical, or topological characteristics (e.g., number of processors/disks, storage locations).	______
o Expandability	o Ease with which the software capability or performance can be increased by enhancing current functions or adding new functions/data.	______

Please estimate the following in terms of manmonths of effort:

	o Effort to expand the software for this application.	______
	o Equivalent effort to do the same task from scratch for this application.	______